Enhancement of low resolution models for real time large scale visualization

Like, Gobeawan

2010

Like, G. (2010). Enhancement of low resolution models for real time large scale visualization. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/20871>

<https://doi.org/10.32657/10356/20871>

ENHANCEMENT OF LOW RESOLUTION MODELS FOR REAL TIME LARGE SCALE VISUALIZATION

LIKE GOBEAWAN

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2010

Acknowledgments

My thesis project would have not been realized now without the guidance and support from all the people around me. I would like to express my gratitude especially to my supervisors Stephen John Turner, Shuhong Xu, and Su Yi. For years, they have never grown tired in guiding me all the way from the start of my PhD to the finishing of my thesis.

My thanks also goes to many friends who have been encouraging me to keep up on my study. Toshi, Anton, and Ratna; I thank them for really sitting down with me many times to listen and to provide feedback to my research work for the sake of my progressing with the work. I also thank my school friends who often drop by to share their study challenges and advices: Stephanus Tandjung, Richard Jayadi, Nganh, Maryam, Chee Kwang, Keisuke, Mobeen, and many more. I would like to thank my sports buddies, Eileen, Yuris, and others for their lively games that recharge me for research.

Many thanks to NTU PDCC laboratory staff and fellow students: Irene, Tay Se, Wang Yong, Yu Hong, and others, for sharing our school burden and spending the longest hours together focusing on our own projects. They have been keeping me going with my research. I wish to thank many NTU staff and lecturers that have been giving their best assistance all along for the progress of my study: Edmond Prakash, who taught me to think freely to create new things, Goh Woon Boon, Francis Lee, Yeo Chai Kiat, Lin Feng, Yap Lee Jong, and many more. They have inspired me to be one of them one day.

I would also like to show my gratitude to many IHPC staff for their technical and

humanity assistance: Zhang Yu, Chiet Sing, Bernard Yeo, Melvin, Kim Huat, Sharon, Kelvin Veragoo, Fu Xiuju, and many more. They have revealed the answer to doing many of the research works, and the happy life of professional researchers. Along with that, I would like to thank A*STAR for sponsoring my PhD study, as well as the A*STAR staff, Santa, Abdul Samir, and many more, that are always ready to answer my curious questions regarding the scholars' welfare.

Finally, my deepest gratitude goes to my family: father, mother, brother, and sister. Thank you so much for always having faith in me, that no matter how I fail, I can always try my best again. Thank you for always listening to me and telling me small advices that actually push me forward to do big things courageously. I dedicate this thesis to my dearest family.

Like Gobeawan

Contents

Acknowledgments	iii
List of Figures	ix
List of Tables	xx
Abstract	xxi
1 Introduction	1
1.1 Framework	3
1.2 Problem Statement	5
1.3 Contributions	6
1.3.1 Seamless Mesh Stitching Using Curve Bones	6
1.3.2 View-adaptive Subpolygon Silhouette Refinement	7
1.4 Thesis Organization	7
2 Literature Review	9
2.1 Handling Large Meshes for Real Time Visualization	9
2.2 Simplification of Large Models	13
2.3 Mesh Stitching	14
2.3.1 Mesh Fusion	16

2.3.2	Hole Filling	17
2.3.3	Surface Reconstruction	18
2.3.4	Curve Reconstruction	19
2.4	Texture Mapping	19
2.5	Silhouette Refinement	20
3	Simplified Models for Real Time Large Scale Visualization	24
3.1	Mesh Cutting	27
3.2	Mesh Optimization	28
3.3	Mesh Stitching	28
3.4	Texture Map Generation	29
3.5	Silhouette Refinement	29
4	Seamless Stitching Using Curve Bones	31
4.1	Objectives	33
4.2	Proposed Concept: Mesh Stitching Using Curve Bones	33
4.3	Seamlessness Criteria	34
4.3.1	Vertex-related criterion	35
4.3.2	Edge-related criteria	35
4.3.3	Face-related criteria	37
4.4	Stitching Methodology	37
4.4.1	Boundary Band Identification	39
4.4.2	Vertex Correspondence	40
4.4.3	Curve Bone Construction	46
4.4.4	Stitch Vertex Approximation	48
4.4.5	Retriangulation	50

Contents	vii
4.5 Experimental Results and Discussion	53
4.5.1 Results	53
4.5.2 Discussion	58
4.6 Contributions	59
5 Adaptive Subpolygon Silhouette Refinement	64
5.1 Objectives	65
5.2 Methodology Overview	67
5.2.1 Silhouette Edge Refinement	68
5.2.2 Subpolygon Silhouette Refinement	70
5.3 Silhouette Edge Refinement	74
5.3.1 Preprocessing	74
5.3.1.1 Feature Extraction	74
5.3.1.2 Approximation of Normals and Tangents	77
5.3.2 Runtime	83
5.3.2.1 Silhouette Edge Extraction	84
5.3.2.2 Estimation of Silhouette Refinement Level	85
5.3.2.3 Calculation of Silhouette Refinement Points	89
5.3.2.4 Silhouette Retriangulation	89
5.4 Subpolygon Silhouette Refinement	91
5.4.1 Preprocessing	91
5.4.2 Runtime	94
5.4.2.1 Subpolygon Silhouette Approximation	95
5.4.2.2 Estimation of Silhouette Refinement Level	98
5.4.2.3 Calculation of Silhouette Refinement Points	100

viii	Contents
5.4.2.4	Silhouette Retriangulation 102
5.5	Experimental Results and Discussion 103
5.5.1	Time Performance 103
5.5.2	Visual Result 110
5.5.3	Discussions 114
5.6	Contributions 118
6	Case Studies 130
7	Conclusions and Future Work 141
7.1	Contributions 142
7.1.1	Seamless Stitching Using Curve Bones 142
7.1.2	Adaptive Subpolygon Silhouette Refinement 143
7.2	Future Work 145
	Appendixes 147
A	Derivation of Subpolygon Silhouette Point Equation 147
B	Supplementary Video 149
C	Author's Publications 150
	Bibliography 151

List of Figures

3.1	Workflow of real time large scale visualization using enhanced simplified polygonal models.	26
4.1	(a) Two half-spheres before stitching, (b) Whole sphere after stitching, showing curve bones and stitch vertices on stitch path.	32
4.2	Workflow of mesh stitching using curve bones: (a) identification of boundary bands (red), (b) vertex correspondence (black dotted lines) with one sample set of vertices for curve bone construction (connected by yellow line segments), (c) construction of curve bones (one sample in orange curve) and their corresponding stitch vertices (one sample in green dot), (d) retriangulation replacing boundary bands with stitch patch (yellow area) around the stitch path (brown line).	39
4.3	Two boundary bands (red bands) of mesh partitions containing boundary vertices (red dots) and near-boundary vertices (blue dots).	40
4.4	Pseudo-code of vertex correspondence rule.	44
4.5	Example of vertex correspondence for unevenly distributed boundary vertices.	45

4.6	A boundary vertex with its corresponding six near-boundary vertices. A representative near-boundary vertex is selected based on dihedral angles of the edges connecting the boundary vertex and its corresponding near-boundary vertices.	46
4.7	Imaginary near-boundary vertex is assigned to correspond to lone boundary vertex.	47
4.8	Bézier curve is constructed from four control points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$	48
4.9	Two-dimensional sketch of boundary bands (green area), a curve bone (red curve) with control points of boundary vertices (blue dots) and near-boundary vertices (yellow dots), and stitch vertices (green dots) lying on curve bone.	49
4.10	Stitch vertices (green dots) are well spaced on the curve bone (red curve) with near-boundary vertices (yellow dots) as the end points, showing automatic blending of resolutions of both mesh partitions (resolutions of both partitions are represented by a simplified view of the boundary band (cyan shaded area).	50
4.11	To replace each boundary vertex $\mathbf{v}_a$ and its surrounding faces, the yellow area is retriangulated with its corners being the near boundary vertices (blue dots) of $\mathbf{v}_a$ and the corresponding boundary vertices of $\mathbf{v}_a$, that is $\mathbf{v}_{b_1}$ and $\mathbf{v}_{b_2}$. The remaining blue area between the two yellow areas is fitted with a triangle.	51
4.12	Potential self-intersection retriangulation arising from a condition where a protruding triangle obstructs the vertex correspondence link between two boundary vertices.	52

List of Figures

xi

4.13	(a) Two simplified partitions of horse model, (b) normals of vertices around boundary bands appear approximately continuous after stitching, (c) seamlessly stitched horse.	55
4.14	(a) Two simplified partitions of camel model, (b) stitched camel shown with its stitch vertices and near-boundary vertices, (c) smooth looking stitched camel, (d) camel wireframe showing seamless stitch result with balanced face distribution around stitch path.	56
4.15	(a) Wireframe of stitched Homer showing natural face distribution around stitch path, (b) close-up image of stitched Homer's nose and mouth area in comparison to directly simplified Homer's in (c) of same resolution.	57
4.16	(a) Two simplified partitions of bumpy sphere model, (b) stitched bumpy sphere shown with its stitch vertices and near-boundary vertices, (c) stitched bumpy sphere, with seamless result along and across the stitch path, (d) wireframe of stitched bumpy sphere, which has seamless stitch path with natural face distribution around stitch path. .	58
4.17	(a) Two simplified partitions of Armadillo model, (b) stitched Armadillo shown with its stitch vertices and near-boundary vertices, (c) wireframe of stitched Armadillo (17532 faces) which has seamless stitch path and natural face distribution around stitch path, similar to wireframe of directly simplified Armadillo of same complexity (17532 faces) in (d).	61

4.18	(a) Two partitions of Homer model with different resolutions: 2658 faces for left partition and 674 faces for right partition, (b) stitched Homer shown with its stitch vertices and near-boundary vertices, (c) wireframe of stitched Homer which shows seamless stitch result across different resolution partitions.	62
4.19	Hausdorff distance comparison of original models, directly simplified models, and stitched models.	62
4.20	Mean distance comparison of original models, directly simplified models, and stitched models.	63
4.21	Root mean square distance comparison of original models, directly simplified models, and stitched models.	63
5.1	Clockwise from top left: original high resolution bunny, simplified low resolution bunny with normal map details yet polygonal silhouettes, simplified low resolution bunny with refined silhouettes (without normal map details), simplified low resolution bunny with normal map details and refined silhouettes.	65
5.2	Workflow of silhouette edge refinement scheme.	69
5.3	Workflow of subpolygon silhouette refinement scheme.	69
5.4	(a) Silhouette edges (red), (b) actual visible subpolygon silhouettes (blue).	71
5.5	Dihedral angle corresponding to an edge in feature check is an angle between two corresponding facet normals.	75

5.6	Vertex color code: black (non-feature vertex), yellow (apex feature vertex), red (valley feature vertex), green (edge feature vertex), blue (corner feature vertex); thick edge lines indicate feature edges. Clockwise from top left, based on feature type of vertex with black border (except black non-feature vertices): apex feature vertex, valley feature vertex, edge feature vertex, corner feature vertex.	77
5.7	Approximation of vertex normals $\mathbf{N}_x$ for each facet x around vertex $\mathbf{v}$ and tangents $\mathbf{t}_i$ for each edge $\mathbf{e}_i$ around vertex $\mathbf{v}$	78
5.8	Approximation of tangent of edges around non-feature vertex.	79
5.9	Smooth edge feature curve is approximated from two feature edges around an edge feature vertex.	81
5.10	Constructing Hermite curve to represent an edge based on positions of vertices and tangents at both ends.	83
5.11	Front-and-back facing test for silhouette edge extraction.	85
5.12	View plane projection of a refinement curve in triangle form.	86
5.13	Upon zooming in to the model, silhouette refinement level adaptively increases, so that the silhouettes still look fine. Grey shaded triangles represent same part of the model. As the model comes closer to the view plane, more polygons “grow” continuously in the direction shown by black arrows.	88
5.14	Silhouette refinement level and spacing between refinement points.	89
5.15	Silhouette retriangulation for silhouette edge refinement scheme.	90
5.16	Centroid of smooth surface formed by three Hermite curves.	91
5.17	Input of Bézier surface parameterization.	93

5.18	Bézier surface control points on a facet (orange plane): 3 vertices (red dots), 9 edge control points and 3 interior control points (black dots).	94
5.19	Two possible scenarios of different visibility status of vertices in a facet (+ is a visible vertex and - is an invisible vertex).	95
5.20	Subpolygon silhouette line connects two subpolygon silhouette points on the subpolygon silhouette edges on a subpolygon silhouette facet. Subpolygon silhouette points have value of 0 in visibility value where their normals are perpendicular to the viewing direction.	96
5.21	Refined feature edge (blue curve) bridges discontinued smooth subpolygon silhouette curves (red curves).	98
5.22	Refined subpolygon silhouette line (red curve) is visually obstructed by edges of the facets it belongs to.	99
5.23	Obstructing edge line is refined along with the subpolygon silhouette line to unblock the view of subpolygon silhouette curve representing the refined silhouette.	100
5.24	Possible arrangements of subpolygon silhouette lines with their barycentric coordinates in a facet.	101
5.25	Silhouette retriangulation for facets containing subpolygon silhouette line.	102
5.26	Frame rates of various models running different silhouette refinement schemes.	105
5.27	Frame rates of running our subpolygon silhouette refinement on various models of different resolutions.	105

List of Figures**xv**

5.28	Performance comparison between subpolygon-silhouette-refined models and PN-triangle-refined models at different resolutions.	107
5.29	Time performance comparison between subpolygon-silhouette-refined knot with higher resolution knots of roughly same silhouette quality without silhouette refinement.	109
5.30	Time performance comparison between subpolygon-silhouette-refined bunny with higher resolution bunnies of roughly same silhouette quality without silhouette refinement.	110
5.31	Time performance comparison between subpolygon-silhouette-refined horse with higher resolution horses of roughly same silhouette quality without silhouette refinement.	111
5.32	Comparison of geometric error of quartic Bézier surface and PN triangle surface of a low resolution hemisphere.	111
5.33	Silhouette quality comparison: (a) 1434-facet knot without silhouette refinement, (b) 250-facet knot with subpolygon silhouette refinement (total 1434 facets).	113
5.34	Silhouette quality comparison: (a) 892-facet torus without silhouette refinement, (b) 300-facet torus with subpolygon silhouette refinement (total 892 facets).	114
5.35	Silhouette quality comparison: (a) 1324-facet bunny without silhouette refinement, (b) 500-facet bunny with subpolygon silhouette refinement (total 1324 facets).	115

5.36	Knot (364 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	120
5.37	Torus (100 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	121
5.38	Bunny (500 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	122
5.39	Horse (600 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	123
5.40	Mannequin (699 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	124
5.41	Armadillo (2000 facets): (a) low resolution with polygonal silhouettes, (b) after subpolygon silhouette refinement, (c) wireframe after subpolygon silhouette refinement.	125

5.42	(a) Low resolution hexagon (6 facets), (b) subpolygon-silhouette-refined hexagon without feature retention, (c) subpolygon-silhouette-refined hexagon with feature retention, (d) low resolution cylinder (86 facets), (e) subpolygon-silhouette-refined cylinder without feature retention, (f) subpolygon-silhouette-refined cylinder with feature retention.	125
5.43	Cylinder (86 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	126
5.44	Vase (610 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	127
5.45	Fandisk (264 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	128
5.46	(a) Low resolution cylinder (86 facets), (b-c) smooth subpolygon-silhouette-refined cylinder viewed from a distance and its wireframe, (d-e) without view-adaptive measure, cylinder from (b) and (c) appear polygonal upon zooming in, (f-g) view-adaptive subpolygon-silhouette-refined cylinder remains smooth upon zooming in.	129

5.47	Extremely low resolution box (12 facets) in smooth shaded and wire-frame modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.	129
6.1	Original large Armadillo (345,944 facets)	130
6.2	(a) Two partitions of large Armadillo (213501 facets for upper partition and 133509 facets for lower partition), (b) two Armadillo partitions are simplified separately into 1230 facets for upper partition and 770 facets for lower partition (total 2000 facets before stitching).	132
6.3	(a) Simplified Armadillo after stitching (2004 facets, 0.6% of original Armadillo), (b) simplified Armadillo after subpolygon silhouette refinement.	134
6.4	(a) Original Armadillo (345,944 facets), (b) Stitched simplified Armadillo (2004 facets) with texture (normal) map, (c) Subpolygon-silhouette-refined stitched simplified Armadillo with texture (normal) map; (d), (e), and (f) are comparisons of zoomed-in around the waist area (stitch area) of (a), (b), and (c), respectively. Silhouette-refined Armadillo with texture map looks like the original Armadillo.	135
6.5	Original large Ramesses (1,652,528 facets)	136
6.6	(a) Simplified Ramesses partitions (3999 facets for left partition and 2999 facets for right partition), (b) Simplified Ramesses after stitching (7028 facets, 0.4% of original Ramesses).	138

List of Figures

xix

-
- 6.7 (a) Zoomed in simplified Ramesses after stitching (7,028 facets), (b) zoomed in simplified Ramesses after subpolygon silhouette refinement, (c) overall simplified Ramesses after subpolygon silhouette refinement. . 139
- 6.8 (a) Original Ramesses (1,652,528 facets), (b) Stitched simplified Ramesses (7028 facets) with texture (normal) map, (c) Subpolygon-silhouette-refined stitched simplified Ramesses with texture (normal) map; (d), (e), and (f) are comparison of zoomed-in chin and shoulder area of (a), (b), and (c), respectively. Silhouette-refined Ramesses with texture map looks like the original Ramesses. 140

List of Tables

5.1	Summary of feature vertex identification scheme.	76
5.2	Summary of normal and tangent approximation.	82
5.3	Comparison of frame rate performance between a silhouette-refined simplified knot (250 facets) with higher resolution knots of similarly- perceived silhouette qualities.	108
5.4	Comparison of frame rate performance between a silhouette-refined simplified bunny (500 facets) with higher resolution bunnies of similarly- perceived silhouette qualities.	109
5.5	Comparison of frame rate performance between a silhouette-refined simplified horse (600 facets) with higher resolution horses of similarly- perceived silhouette qualities.	110

Abstract

The latest data acquisition technology enables us to obtain huge 3D model data. To visualize the large 3D models in real time, large memory resources are required. However, an off-the-shelf PC often has limited memory compared to the size of such large models, so that the computer cannot display the models at an interactive or real time frame rate. Real time visualization of a large model on a limited memory system can be realized using a simplified version of the model. This thesis discusses such a framework, with two main contributions in the areas of mesh stitching and silhouette refinement.

In a complete framework using the limited available memory, the large scale model is cut into several partitions, and they are then optimized (simplified, remeshed, and so on) separately. The outcome is some optimized low resolution mesh partitions with mismatched boundaries. To join the partitions together into one simplified model, this thesis proposes a seamless mesh stitching method using the curve bones, which are obtained from the boundary correspondence guided by a set of seamlessness criteria. Analogous to stitches in sewing clothes, curve bones join two mesh partitions while retaining the features across the boundaries. Stitch vertices found on the curve bones replace the boundary points. The boundaries are retriangulated to accommodate the stitch vertices. The result is a seamless stitch that fills the gap and blends the resolutions between the partitions. The stitched simplified model is then ready for visualization.

The appearance of the simplified model can be enhanced by adding texture details such as surface normals, bumps, and so on to the simplified model polygons. However,

it still lacks visual appeal due to the polygonal silhouette artifacts. This thesis presents a method to refine the silhouettes in real time with smooth transition for any changing view point. The subpolygon silhouette segments of a model are refined into silhouette curves which lie on the preconstructed smooth Bézier surfaces of the model without using the original model as a reference. The refinement level is view-adaptive. The method preserves the features of the model and produces consistent silhouette shapes based on the preconstructed Bézier surfaces. The refined silhouettes are two-manifold with the models.

Overall, the thesis shows that large models can be represented realistically by enhanced simplified models for real time visualization with limited computer resources. Large models are cut and simplified/optimized separately, and then are stitched back seamlessly as whole simplified models. The simplified models are enhanced in their appearance by adding texture details and refining the polygonal silhouettes during visualization.

Keywords: real time visualization, 3D large polygonal models, simplified models, seamless mesh stitching, polygonal silhouette refinement, visual realism.

Chapter 1

Introduction

The latest digital acquisition technology enables us to scan various real world 3D objects into high density digital data representing 3D models. The exponentially increasing size of the acquired data has outgrown the available size of computer memory for processing, while the need for real time visualization of such large models has been increasing for applications in areas such as scientific simulation, art and historical simulation, 3D games, medical visualization, and many others. Even though the huge size of the data makes it possible to view the 3D model at a highly detailed level, it is often too big to fit into the available memory of an off-the-shelf home PC.

Real time visualization of large polygonal models often requires the model geometry data for various visualization tasks such as color encoding (commonly used in scientific visualization), collision detection, and lighting. This requires an equivalently large amount of memory to load the model data in order to visualize the large model in real time. As the available memory is scarce relative to the size of the model, real time visualization of such large polygonal models using most existing ‘in-core’ visualization methods is near to impossible, unless some additional optimization work is

done.

Although there is no specific standard technical specification to define “a resource-limited computer system”, the author took as an example of specification, one readily available home PC with 2 GB random access memory and 256MB graphics memory, which is a common specification of home PC sold in the market nowadays. A system with such resources is limited in its ability for real time visualization of large models, as shown in the case study in Chapter 6. The scanning technology of today is able to produce large models with sizes above hundreds of million polygons. It requires more than today’s off-the-shelf PC random access memory of 2 GB in order to load such a large model into the memory. Even if the large model can fit into the memory, additional memory is still required to process the data and to provide real time interaction with the model.

Theoretically, a PC with 2 GB random access memory should be able to load around four hundred million polygons without the presence of other tasks that use the random access memory. In practice, existing commercial software tools can only, in general, load up to a million polygons before running out of memory. Loading larger models will crash the programs. Some explanations can be offered. The PC operating system may be occupying the random access memory for some essential basic services, or the visualization tool itself requires a large amount of random access memory to run. In many cases, upon loading the model into the data structure of the visualization program, a lot of additional information is being generated and processed. The amount of runtime data being loaded to the random access memory for runtime processing, or to the graphics memory (256 MB) for display, may overflow and make the program crash. This scenario represents the practical situation of how

a PC with 2 GB random access memory can handle in reality far fewer polygons than the theoretical number due to various factors in the PC and programs.

The ‘in-core’ approach of level-of-detail using streaming pipeline and multi-thread processes on a graphics card is only possible for models that can be processed in-core (models that can wholly loaded into the available random access memory). If the size of the large model is too big to fit into the random access memory, the real time performance of any ‘in-core’ approaches will eventually be inhibited by expensive disk operations of transfer of the polygon data between disk memory and random access memory.

With the limitation of available memory resources for real time visualization of large models as described above, the most common practice adopted in computer graphics is to cut the large models into partitions of manageable sizes, to simplify and optimize each partition separately, and then to join the simplified partitions into a simplified model. However, many of the visual details are lost in the simplification process. As we explore a framework for real time visualization of large polygonal models using simplified models in this thesis, there are several potential areas in which the framework can support improvements to their visual quality. The thesis explores efforts for enhancing the simplified models in the whole framework of real time visualization of the large polygonal models.

1.1 Framework

A framework of using simplified polygonal models to replace the large models for real time visualization is constructed by adapting existing work on large model compression, texture-based simplification, and real time visualization of low resolution

polygonal models.

For large model compression, works such as Bernardini *et al.* [BRM*02] and Ho *et al.* [HLK01] adopt an approach that cuts a large polygonal model, which cannot be loaded all at once into the memory for simplification (i.e., cannot be simplified ‘in-core’ directly), into several partitions that can fit into the memory. The partitions are simplified separately using certain ‘in-core’ simplification, then are stitched together to form a whole model again. The suggested mesh stitching is a straightforward connection, assuming that the preceeding simplification retains original inter-partition-boundary geometry connectivity information. In practice, not only is simplification needed, but remeshing or some other optimization to the model is done. In addition to the simplification that retains the boundary, there are various versatile ‘in-core’ mesh simplification methods for different purposes. Hence, in our framework, we assume a generality that the mesh partitions may undergo some geometry processing not limited to a particular method of simplification. Our general framework thus includes mesh stitching with the general condition of no knowledge of the connectivity of the partition boundaries.

In the research topic of mesh simplification, there are many existing classic and versatile texture-based simplification and texture mapping methods. These methods have shown excellent results for providing inner details on the simplified polygons that resemble the original large models, as discussed in Chapter 2.

During rendering of the simplified polygonal models with their inner texture details, the simplified models appear polygonal along the silhouettes. While several works have been done to refine these silhouettes, there is still room for improvement, especially for the accurate smooth transition of refined silhouettes of low resolution

polygonal models.

The thesis presents two main contributions in the whole framework of using simplified polygonal models to achieve real time visualization of large models: a seamless mesh stitching method using curve bones and a view-adaptive subpolygon silhouette refinement.

1.2 Problem Statement

Current computer graphics practice uses simplified models to replace the large polygonal models in real time visualization on a system with limited resources. In general, the thesis aims to enhance the simplified models to simulate the large polygonal models.

In a practical scenario, large polygonal models are too big to fit into the available memory even for simplification purposes. Hence, before the simplification, a large model is cut into several partitions of sizes that fit into the memory. After the simplification of the partitions, they are joined together (stitched) at the boundary. One potential problem arising from the stitching process is the visible seams as a result of the stitching. As there are not many existing works which attempt to stitch mesh partitions seamlessly, **this thesis seeks for a new stitching method that effectively joins two partitions at their boundaries seamlessly with minimal change to the existing geometry and topology.**

After the simplified partitions are stitched together into one simplified model to represent the large polygonal model, the simplified model can be used for real time visualization in a system with limited memory resources. Simplified models, however, are lacking in visual details compared to the original large models.

The lack of visual details are evident in the absence of details on the surfaces of the simplified polygons compared to those of the original model. Existing studies suggest various robust texture mapping methods to paste texture details onto the surface of simplified polygons. However, even with the inner details, a simplified model is fundamentally a low resolution model with polygonal silhouettes. This polygonal silhouette shape distinguishes the simplified model from the original high resolution model. **This thesis aims to enhance the appearance of the simplified polygonal model with texture details by refining its polygonal silhouettes for real time performance.**

1.3 Contributions

This thesis contains two main contributions in the overall framework of using simplified polygonal models for real time large scale visualization. They are:

1. seamless mesh stitching using curve bones
2. adaptive subpolygon silhouette refinement

Each proposed method and its sub-contributions are highlighted as follows. The details of the contributions of this thesis are described in Chapter 3, Chapter 4, and Chapter 5.

1.3.1 Seamless Mesh Stitching Using Curve Bones

Given the inputs of two mesh partitions with fixed topology and orientation but mismatched boundaries (assuming no prior knowledge of connectivity between the boundaries), the proposed seamless mesh stitching method joins the partitions to-

gether by using curve bones that stitch across the boundaries along a stitch path. This is analogous to sewing two pieces of clothes at their boundaries. After each boundary of each mesh partition is identified, the points along the boundaries are paired up according to a correspondence rule which decides the position of each stitch vertex across the boundaries. Appropriate curve bones are then constructed, and the boundaries are then modified to accommodate the stitch vertices on the curve bones, producing a seamlessly stitched mesh.

1.3.2 View-adaptive Subpolygon Silhouette Refinement

To refine the polygonal silhouettes of a low resolution polygonal model during the runtime stage of visualization, the preprocessing tasks of feature extraction, geometry attribute approximation, and surface parameterization of the polygonal model are done. The information obtained in the preprocessing is then used during runtime for silhouette extraction, silhouette refinement level decision, silhouette refinement point calculation, and silhouette retriangulation. The polygonal model is rendered with visibly fine silhouettes. When the view point moves, the fine silhouettes will adapt smoothly with consistent shape.

1.4 Thesis Organization

The remaining chapters of this thesis are organized as follows.

Prior to explaining the framework of real time large scale visualization using simplified models, Chapter 2 summarizes the existing related works and discusses some basic concepts required to better understand the contributions described in this thesis. This chapter explores various state-of-art large scale visualization works, various

polygonal model simplification methods, and the related works in areas of mesh stitching, texture mapping, and silhouette refinement.

Chapter 3 describes the whole framework of real time large scale visualization adopted in this thesis. The whole framework potentially realizes the real time visualization of large polygonal models on a resource-limited computer system using the simplified models.

The next two chapters elaborate on the main contributions of the thesis. Chapter 4 explains in detail the first contribution on a seamless mesh stitching method. Chapter 5 details the second contribution on silhouette refinement of low resolution polygonal models. The following Chapter 6 presents two case studies of applying the two methods within the complete framework to large models in order to achieve real time visualization.

Finally, the thesis is closed with conclusions and future work in Chapter 7, summarizing the main contributions of this thesis and the potential future work.

Chapter 2

Literature Review

This chapter discusses existing work in relation to large scale real time visualization. Works relevant to the proposed methods in this thesis - mesh stitching and silhouette refinement - are also discussed.

2.1 Handling Large Meshes for Real Time Visualization

According to the surveys by Silva *et al.* [SCESL] and Isenburg *et al.* [ILGS03], there are three common approaches in handling (processing) large polygonal meshes for visualization on a limited memory system: batch processing, online processing, and mesh cutting.

Batch processing streams large meshes into the random access memory in batches of polygons. Each batch of polygons is then processed sequentially in the random access memory. To use the batch processing approach to handle a large model, the file content of the large model needs to be in polygon soup format. In general, a

polygonal mesh file is available in vertex-indexed polygon format, not polygon soup format. A polygon soup is a group of polygons with unknown relationship among the polygons due to unindexed vertices (unknown in terms of which vertex is topologically shared by which polygons; the vertices of each polygon are independent of those of other polygons).

In order to convert indexed polygons into polygon soups, the large model mesh has to be read in streams through several time-consuming passes. The polygon soups occupy twice the space of indexed meshes, and the vertex dereferencing of indexed meshes to polygon soup format is computationally expensive. Even though the polygon soups can then be displayed more quickly on the PC by streaming them from the disk memory to the random access memory without expensive and repetitive dereferencing of vertex indices from the disk memory, further processing of and interaction with the large model may involve a global search operation on all polygon soups. This search operation potentially causes memory thrashing, because it is an expensive I/O operation that requires intensive access to the disk memory. Thus, well-established simplification and optimization methods, which are mostly ‘in-core’ methods that require all the input data to be loaded and processed at once in the random access memory, are not suitable for use in the batch processing approach. The limited number of visualization methods that can be used in the batch processing approach implies limited possibilities in processing the large models to produce the desirable result.

Such an approach is adopted by the works of Lindstrom [Lin00], Lindstrom and Silva [LS01], Shaffer and Garland [SG01], Garland and Shaffer [GS02], and Isenburg *et al.* [IG03]. For instance, in the work of Lindstrom [Lin00], large polygonal models are simplified by using a vertex clustering method to merge all vertices in the same

cell of space.

Using the online processing approach to handle the large model, preprocessing is done to index the polygons and organize the indexing into some structure, for example, an octree structure, for fast random access during runtime. While the main polygon data are kept in disk memory, the much more compact index information is loaded in random access memory for fast access to the polygon data. However, in visualizing the large model for real time performance, despite the fast indexing to the polygon data, continuously retrieving the polygon data from disk memory is a very costly operation. Similar to the batch processing approach, using ‘in-core’ methods in the online processing approach potentially causes memory thrashing. Hence, the online processing approach is not suitable for most well-established ‘in-core’ simplification and optimization methods.

The online processing approach is adopted in the works of Levoy *et al.* [LPC*00] and Cignoni *et al.* [CMRS03]. In the latter work, the large mesh is reorganized into a compact octree-based structure which contains access indices to the large mesh stored in the disk memory. During runtime, the octree-based structure in the random access memory is used to query the large mesh data from the disk memory.

The mesh cutting approach, on the other hand, is simpler to implement. In the mesh cutting approach, the large mesh is cut into several mesh partitions such that each can fit in the available memory. The partitions are each processed separately while retaining the cut boundaries for easy rejoining at the boundaries subsequently. The processing can be simplification, compression, remeshing, and other optimization steps. This is followed by stitching the partitions together into one simplified model. This simplified model fits into the main memory for real time visualization.

The mesh cutting approach for handling large meshes has been widely adopted by various works such as those of Hoppe [Hop98], Prince [Pri00], Ho *et al.* [HLK01], and Bernardini *et al.* [BRM*02]. This approach is the most commonly adopted approach, because mesh partitions that fit in the available (main) memory can be processed with practically any ‘in-core’ methods that have been developed so far. Existing work, however, tends to use only certain ‘in-core’ methods that retain the partition boundaries for easy and direct stitching of the partitions.

For instance, Bernardini *et al.* [BRM*02] build a large 3D polygonal model of Michelangelo’s Florentine Pieta by cutting the acquired large model into parts and simplifying them in-core in two passes. In the first pass, the mesh partitions are simplified in-core while leaving the boundaries intact, and then are joined together at the boundaries directly. The second pass involves cutting this simplified model into two other parts, simplifying the partitions in-core (to simplify the part which has not been refined in the first pass) while retaining the current boundaries again, and stitching the partitions again to yield a whole simplified model.

Works of Sander *et al.* [SSGH01, SM06] partition a large model into several charts based on the planarity and compactness (chart perimeter). A chart is a cluster of polygons of the large model which are considered homogenously planar; a chart corresponds to a texture map. These charts can then be processed separately (in these works, it is for the texture mapping process). The mesh cutting method can adopt the same principle of the proposed chart partitioning. By only cutting the model exactly at the chart boundaries, the mesh partitions can be directly reconnected together later with only minor modification to the texture maps at their boundaries. This works only with the assumption that different charts are discontinuous at their

boundaries (visible boundary path on the whole model before cutting). In this case, a direct stitching (without any smoothing or modification) of two partitions at their boundaries might be acceptable even if the stitch is visible (not seamless).

Unlike existing work, this thesis adopts the mesh cutting approach assuming that the partition boundaries are arbitrary and potentially mismatched, due to the separate simplification, remeshing, or other ‘in-core’ optimization processes. This mesh cutting approach enables the use of all well-established optimization and simplification methods, including the ‘in-core’ methods that are not suitable for use in the batch processing and the online processing approaches.

In conclusion, for real time visualization of large models on an off-the-shelf PC that flexibly supports the use of any existing ‘in-core’ simplification and optimization methods, the mesh cutting approach is suitable and relevant. While this approach makes real time visualization possible, there are still problems of loss of quality in the large models. This thesis thus presents the novel seamless mesh stitching and silhouette refinement methods to complement the mesh cutting approach in enhancing the models in a large scale visualization framework.

2.2 Simplification of Large Models

In view of the large size of the polygonal model that cannot fit into the memory, there have been lots of work done to compress or simplify the models. Luebke [Lue01] and Cignoni *et al.* [CMRS98] extensively survey and compare simplification methods for various purposes. One of the widely used simplification methods is the work of Garland and Heckbert [GH97] which simplifies polygonal models to any desired resolution using quadric error metrics.

Many efforts have also focused on producing compact and efficient mesh representations. Hoppe [Hop96] proposed a progressive mesh representation which represents a polygonal model in a lossless multiresolution form, while Rusinkiewicz and Levoy [RL00] represent a polygonal mesh in the form of a more compact point cloud rather than polygons. Such representations enable faster rendering of the large meshes for certain display purposes.

On the other hand, the simplification method is often combined with some optimization methods for handling large scale visualization, in which the system handles a large number of objects in a scene while the user explores around the scene. Aliaga *et al.* [ACW*99], for example, handle large models or scenes using a framework of level-of-detail and image-based 3D visualization. Large models or scenes are rendered in scene-graph fashion, using level-of-detail for objects inside the view frustum and image texture impostors to replace the background. As the viewpoint moves in the scene for exploration, various cullings are done to speed up the scene rendering. The proposed framework in their work might be memory intensive for the runtime prefetching part and not suitable for complex scenes without more optimization work.

2.3 Mesh Stitching

Mesh stitching is a process of joining two meshes together at their boundaries, retaining the topology and geometry of the meshes as much as possible. In general, mesh stitching involves the tasks of

- *positioning*, that is the establishment of orientation and correspondence between boundaries of the two objects to be stitched together,

- *merging*, that is the establishment of a suitable stitch path (merging points) on the two objects, and
- *smoothing*, that is the modification of the object surfaces around the stitch path to join the two objects together

Turk and Levoy [TL94] propose a stitching method for two aligned meshes by removing the overlaps at the boundaries and directly “zippering” the boundaries by tessellating the gap between the boundaries. The visible stitch is corrected by modifying the stitch vertex positions along the surface normal direction according to the average positions of corresponding points on the original mesh partitions. This operation requires processing of original reference models which may not be readily available or may be too large for use in a system with limited resources.

Gueziec *et al.* in [GTLH01] suggest the method of snapping (merging) pairs of stitchable edges from two meshes. The edge pairs are stitchable if they are relatively close in distance. The work does not automatically consider the smoothness around the merging area, hence a post-processing step of smoothing is required for a seamless result.

Existing mesh stitching work have not thoroughly benchmarked their performances nor emphasized much on the seamlessness result to retain the geometry (and features) as much as possible. Hence, this thesis presents an automatic seamless mesh stitching using the lightweight curve bones, together with feature retention. More details are presented in Chapter 4.

Although mesh stitching is not specifically well researched, it is related to existing work in other areas. The following reviews the literature of relevant areas such as mesh fusion, hole filling, surface reconstruction, and curve reconstruction.

2.3.1 Mesh Fusion

Another area closely similar to mesh stitching is mesh fusion. Mesh fusion is a process of blending two meshes together at any desired merging points (not limited to boundary merging as for mesh stitching), usually involving the geometry and topology modification of the two meshes for blending into one unit. Mesh fusion is often the act of cut and paste (may be followed by smoothing or blending) of one mesh onto other mesh surfaces.

Kanai *et al.* [KSMK99] join two meshes together at the boundaries formed by several vertices specified by the user. This method automatically adjusts the orientation of both meshes to fit at the boundaries by rigid transformation (rotation and translation) and scaling of one mesh. To complete the fusion process, this transformation is propagated from the boundaries to the whole mesh by deformation. The mesh fusion is controlled by a function that combines the vertices of this mesh with the vertices of the other mesh. This method requires the user to define a certain mathematical function to achieve seamless fusion at the boundaries, which is neither intuitive nor automatic. The method deforms the whole mesh partition in order to fit smoothly with another mesh partition.

Similarly, Yu *et al.* [YZX*04] merge two meshes at their different boundaries by adjusting the boundaries to one intermediate boundary. The boundary change is propagated to the inner parts of the meshes by deformation (transformation and smoothing). Some user interaction is required to determine the vertex correspondence at the boundaries.

Sorkine *et al.* [SCOL*04] and Alexa [Ale06] suggest merging two meshes at the boundaries using a Laplacian surface editing method, with the initial condition of

having the same number of vertices at both boundaries for one-to-one mapping (pre-processing of remeshing may be done to achieve this condition). After simply zipping both boundaries, a transitional region is selected and its resolution is obtained by the interpolation of the resolutions of both sides.

Jin *et al.* [JLW*06] introduce the concept of fusing meshes at any number of boundaries (openings) instead of the usual one-to-one fusion at a pair of boundaries. It achieves this by using two offset planes containing any number of boundaries, in between which the intermediate surface is generated by a certain function to seamlessly connect all the boundaries to produce one fused mesh.

Mesh fusion is different from the mesh stitching concept in terms of the constraints. Mesh stitching does not involve deformation of the meshes to be joined together. In this thesis, mesh stitching aims to join two meshes together at the boundaries without any change to the non-boundary parts of the meshes and with minimal change to the boundaries. The mesh stitching method assumes that existing mesh partitions are complete and do not require unnecessary additional filling.

2.3.2 Hole Filling

Hole filling is common in the field of CAD model repair. The hole filling method closes holes or gaps in polygonal models. Some existing works in hole filling are those of Barequet and Sharir [BS95], Pfeifle and Seidel [PS96], Barequet *et al.* [BDK98], Borodin *et al.* [BNK02], Borodin and Klein [BK02], Liepa [Lie03], Bischoff *et al.* [BPK05], and Chong *et al.* [CKL07]. Other related model repair work include those of Butlin and Stops [BS96], Steinbrenner *et al.* [SWC00], and Petersson and Chand [PC01].

Hole filling is similar to mesh stitching in that one of the aims of both is to create a continuous surface between two boundary-bounded surfaces. In general, both hole filling and mesh stitching methods are initialized by the correspondence between vertices to be patched or connected together and are finished by some tessellation around the stitch or hole. While the mesh stitching method joins the boundary edges together by removing the overlapping boundaries and filling the gaps (if necessary) and zipping the boundaries, the hole filling method connects the boundaries by filling in the gap between the boundaries.

2.3.3 Surface Reconstruction

In general, surface reconstruction generates 3D surface patches from a group of unorganized points. There are several categories of surface reconstruction methods, and one of them is by joining together some parallel contours to form a complete mesh model. The adopted method in joining these contours can be adapted to the general purpose mesh stitching.

The early surface reconstruction work [FKU77,Boi88,dCS96,OPC96] focused on problems of correspondence and retriangulation in between the contours to produce a complete polygonal mesh.

Other works [MSS92,PK96,BST00,NBM05] introduce the methods of fitting the smooth surfaces to the contours to construct the polygonal mesh. The surface fitting methods are discussed in detail in [Die93] and [WARV02].

2.3.4 Curve Reconstruction

From a set of known points, a smooth curve can be constructed. The constructed curve can either interpolate (closely fit) or approximate the points. Well known curve reconstruction methods include those of Hermite curve, Bézier curve, and B-spline. Curve reconstruction or curve fitting is discussed extensively in [Die93] and [Sal05].

In the work proposed in this thesis, instead of using a computationally expensive surface fitting approach, a curve approximation method is developed for the mesh stitching. It is mathematically less complex as it involves a one-dimensional calculation (one parameter) compared to the surface fitting approach, which requires an at-least two-dimensional calculation (two parameters). Smoothness and features along and across the stitched path are not severely compromised by using curve approximation because seamlessness criteria are considered in stitching procedure, as elaborated in Chapter 4.

2.4 Texture Mapping

Simplified models typically lack surface details. More details can be added by re-increasing the number of polygons (by subdivision, higher level-of-detail progressive mesh [Hop96]). However, increasing the number of polygons slows down rendering performance relatively significantly and defeats the purpose of large mesh simplification for real time rendering.

Instead, a texture mapping method is adopted. Two dimensional images containing the surface details are pasted onto the surface of simplified models. The system continues to render the same low number of polygons. Consequently, simplified models with texture details can be displayed similarly to the original large models without

affecting the real time rendering speed too much.

A texture map may contain various surface attributes such as color, normal, curvature, displacement, and bumps. The texture is generated by sampling the attribute values of the original large models and storing them in an image map. The information on the texture image is then mapped onto the simplified polygon surfaces.

Several works such as [COM98,SSGH01,SM06] tackle the texture generation and mapping issues together with the model simplification process. This approach ensures that the model is simplified while generating texture with minimal deviation (stretch/strain) from the original model. These methods restrict the kinds of simplification methods that can be used together with the texture mapping. Another approach of texture generation and mapping such as those of Cignoni *et al.* [CMSR98] works on any general simplified models by sampling the attributes of original models and projecting them onto the points of their simplified versions.

2.5 Silhouette Refinement

A silhouette is an important aspect of identifying an object, and a lot of work has been done mainly to extract the silhouettes of models for various purposes such as non-realistic rendering applications, shape recognition, and so on. Some works focus on refining polygonal silhouettes of low resolution models for better visual quality and accuracy. The extensive surveys for silhouette extraction methods by Wang *et al.* [WTD04] and Isenberg *et al.* [IFH*03] recommend various silhouette extraction methods for different purposes.

In order to refine the silhouette of a polygonal surface, the geometric attributes of the vertices which define the smooth surface are used in the calculation. They are

mainly the normal and tangent vectors of the vertices. These attribute values can be estimated using one of many methods as surveyed by Petitjean [Pet02] as well as Garimella and Swartz [GS03]. Max [Max99] proposes vertex normal estimation by face-area-weighted averaging of the normals of corresponding faces around the vertex, while Chen and Wu [CW04] use centroid distances from the vertex as the weights in averaging the normals.

The subpolygon silhouette extraction method proposed by Hertzmann and Zorin [HZ00], is especially related to the topic of silhouette refinement. Instead of extracting silhouette facets/edges/vertices like most of other works suggest, Hertzmann and Zorin suggest to extract the silhouette subpolygon lines. Silhouette subpolygon lines are the lines inside the polygon containing all the surface points which are the actual silhouettes, as their normals are approximately perpendicular to the viewing direction. These subpolygon silhouette lines, instead of the silhouette facets and silhouette edges, are the actual lines visible as the silhouettes. Detecting subpolygon silhouette lines ensures accurate silhouettes for visually smooth transition of silhouettes upon the changing of view point. The silhouettes also look more refined than those produced by other silhouette extraction methods. However, the refinement is not effective enough for low resolution models with severe polygonal silhouette artifacts.

Wang *et al.* [WTW*08] take this method one step further to do the real refinement on the subpolygon silhouettes to improve their earlier silhouette edge refinement work [WW03]. They suggest to construct a smooth curve on top of each silhouette subpolygon line and then to paste a triangle strip connecting the curve and the subpolygon line onto the facet creating non-manifold part, such that the model appears to have fine silhouettes from the view point. While the silhouette refinement result is

excellent for fast viewing and smooth transition of silhouettes, the proposed curve construction method is based on the constructed curve of two siding edges on a three-edge facet. It provides no guarantee for close-fitting, consistent shape of the silhouettes upon changing view point, because consistent curves on one smooth surface can only be constructed from the surface parameters of three edges, instead of two edges, of a facet. In order to show the smooth silhouettes, the blocking vertices are perturbed, which in fact, changes the very basic geometry of the models. The non-manifold nature produced by the refining polygons on top of the model surface may not be suitable for collision detection in 3D space, where it is possible that in a multi-object scene, the refining silhouette strips of a model overlap with the boundaries of other objects. This thesis therefore proposes a different subpolygon silhouette refinement method to solve the above problems, while maintaining real time performance.

The other closely related works are those of Vlachos *et al.* [VPBM01] and Dyken *et al.* [DRS08], who both refine the silhouette faces using the subdivision concept. The two works provide a smoothly transitioning silhouette given a changing view point.

Classic works of Hoppe [Hop96], Prince [Pri00], and Sander *et al.* [SSGH01] naturally produce view-adaptive high level of details near the silhouette areas, with the constraint that it applies only for progressive mesh presentation. Nevertheless, the resolution of the fine silhouettes is limited by the highest available level of details of the model. The silhouettes may still appear polygonal upon zooming in past the highest level of details.

Some other silhouette refinement works, such as Gu *et al.* [GGH*99] and Sander *et al.* [SGG*00], propose silhouette clipping methods yielding fine silhouettes in the image space by referencing the fine silhouettes of the original models during runtime.

While the refined silhouettes are accurate and relatively fast, the methods only refine external silhouettes (outline of the shape) and require the original models during runtime as reference. They also render the models in inflated volumes. Similar to these works, Wu *et al.* [WWWG06] store the fine silhouette information in the form of silhouette texture maps which are used to “carve” out (skip the rastering for) the inflated models according to its current silhouettes, in order to produce fine silhouettes in the image space.

Works of Wang *et al.* [WTW*08], Dyken *et al.* [DRS08], and Wu *et al.* [WWWG06] are further optimized by making use of the GPU in the graphics pipeline for fast rendering performance of large models. The silhouette refinement approach presented in this thesis shows that an alternative efficient method can still perform the task of acceptable real time visualization of silhouette-refined low resolution models even before GPU optimization. This is due to efficient refinement only around silhouette-related edges instead of refining the whole surface patch around the silhouettes.

Another image space silhouette refinement method proposed by Foster *et al.* [FSSW07] smoothens the silhouette lines on the display screen by reverse subdivision for a non-realistic rendering purpose. Such an image space method does not refine the polygonal silhouettes in 3D space.

Based on the existing work, this thesis proposes a novel method to refine the polygonal silhouettes (both internal and external silhouettes) into 3D smoothly transitioning fine silhouettes as an integral part of the two-manifold models, as detailed in Chapter 5. The proposed method retains the model features and refines polygonal silhouettes adaptively for possibly infinite levels of refinement in real time.

Chapter 3

Simplified Models for Real Time

Large Scale Visualization

This thesis presents a framework of real time large scale visualization, adopting various existing works and complementing them with the new methods. The objective of the framework is to visualize large 3D polygonal models in real time for good visual results on computer systems with limited memory.

As mentioned in Chapter 1, the memory capacity limits the real time performance of rendering the polygons of the model. Also, the more polygons to be rendered, the more time the system needs to finish displaying one frame of display on the screen. For real time visualization, the number of frames to be displayed one after another in one second, that is the frame rate, is set as a standard of minimum 25 fps (frames per second). At this rate and above, the audience will experience smooth animation of the visualization without any detectable screen flicker.

To display the large polygonal model in real time with the limited memory, the task of rendering many polygons which consumes significant computing time can be

Chapter 3. Simplified Models for Real Time Large Scale Visualization 25

reduced by simplifying the large model. The loss of the visual quality of the simplified model can then be compensated by the lighter tasks of enhancing the appearance of the simplified model, while retaining the real time performance of the system.

The adopted framework in this thesis uses the mesh cutting and stitching approach in handling the large meshes. This general framework is packaged to be as versatile as possible - assuming flexibility of choosing existing methods to suit a particular need for each task in the framework. The large mesh is cut into several partitions that can each fit in the available memory, so that they can be processed separately within the system memory capability. The processing can be simplification, compression, remeshing, and other optimization steps. Thereafter, the simplified partitions are stitched together into one simplified model and then visualized in real time.

The processes in the whole framework, as illustrated in Figure 3.1, are divided into two main parts. The first part, handling the large models, consists of processes outside the dotted box, which convert the large meshes into simplified ones and their detail maps. The second part, the visualization, inside the dotted box, visualizes the simplified models. The explanation of the figure is as follows.

A large polygonal model is cut into n partitions that can fit individually into the available memory. Each partition is processed separately for optimization, producing simplified partitions. The simplified partitions are then stitched together into one simplified model. Each original partition is then sampled to generate one part of the whole texture map for the simplified model. The outcome of the texture map generation is a whole texture map wrapping the simplified model. A simplified model together with its texture map is then ready for real time visualization.

After some preprocessing, the simplified model is visualized with its silhouettes

26 Chapter 3. Simplified Models for Real Time Large Scale Visualization

Figure 3.1: Workflow of real time large scale visualization using enhanced simplified polygonal models.

being refined. The texture map to be pasted onto the simplified model is adjusted according to the silhouette refinement. The final rendering on one screen frame shows an enhanced simplified model. By using the enhanced simplified model, visualization of the original large model can be achieved in real time.

The main contributions of this thesis are highlighted in grey in Figure 3.1, that is in the areas of mesh stitching and silhouette refinement. As this thesis adopts the most common practice of handling large meshes, that is the mesh cutting approach, novel methods in mesh stitching and silhouette refinement are required to solve problems arising from the adopted approach. These problems appear, because the adopted approach is general and flexible without many constraints of use of certain methods or representations as suggested by existing works, as discussed in the previous chapter. The proposed methods of mesh stitching and silhouette refinement in this thesis aim to complement other processes in this framework. The following sections discuss the roles of various processes including the two main contributions in the overall framework.

3.1 Mesh Cutting

The adopted complete visualization framework allows flexible use of existing methods to process the meshes. For example, instead of choosing certain methods in order to retain the boundaries during the mesh cutting and optimization processes for the sake of straight forward direct stitching, the adopted framework does not restrict the choice of methods. A large polygonal model, according to a specific need, can be cut in various ways by traditional space segmentation or modifying some existing ‘in-core’ cutting methods such as [MW99, FKS*04, LLS*04, LSTS04, LZHM06] for ‘out-of-core’ application.

Traditional space segmentation can be adopted by streaming the large model into the limited available memory and segmentating the model based on the space each element of the model occupies. The segmentation may involve several passes to categorize all points of the model into several partitions.

3.2 Mesh Optimization

Following the cutting, the partitions are then optimized independently using any existing methods, depending on the specific need. Optimization includes processes such as simplification, compression, remeshing, smoothing, etc. Existing simplification methods, for example, have been discussed in Section 2.2. The optimization processes may modify the model partitions without retaining the topology and geometry of the boundaries.

3.3 Mesh Stitching

After the optimization, the mesh partitions have to be stitched together at the boundaries to form a whole simplified model. Due to the flexibility of the individual processes of mesh cutting and optimization in this framework, the boundaries may have been modified independently, resulting in the mismatched boundaries. Stitching the partitions together at the boundaries becomes nontrivial. A mesh stitching that produces a seamless result is required. As discussed in Section 2.3, existing mesh stitching methods have not addressed the seamlessness problem in detail. Hence, this thesis presents a novel seamless mesh stitching method. This contribution will be discussed in detail in Chapter 4.

3.4 Texture Map Generation

The simplified model, despite its small affordable size for real time visualization, lacks the visual detail compared to the original large model. It can be enhanced by mapping detail texture on its surface. The texture map for the simplified model is generated by extracting the surface details such as vertex normals and vertex colors from the original model partitions. The map contains details of the surfaces of the simplified polygons that portray the actual surface appearance of the original model. The simplified model data and the texture map combine to show the correspondence between the simplified model vertices and their normals and or colors of the original model. The choices of various texture map generation methods are discussed in Section 2.4.

3.5 Silhouette Refinement

Even though the simplified model with the texture details on its surface are very similar to the original model, it still has polygonal silhouette artifacts. Human vision is sensitive to silhouettes of an object in perceiving the shape of the object. The polygonal silhouettes make the object look less realistic. To enhance the textured simplified model, the polygonal silhouettes are refined during the visualization runtime.

The existing silhouette refinement works, as discussed in Section 2.5, still lack one or more of the desired characteristics such as smooth silhouette transition, consistent silhouette shapes, two-manifold result, feature retention, real time performance, and view adaptiveness. These characteristics of the silhouette refinement method are required for the desired silhouette refinement result in the adopted visualization framework. This thesis presents a novel silhouette refinement method, with all of

these characteristics, that refines the silhouettes of a model adaptively and smoothly in real time despite the changing of view point, while preserving the actual shapes and features of the model. The contributions of the silhouette refinement are discussed in detail in Chapter 5.

Chapter 4

Seamless Stitching Using Curve Bones

In the real time large scale visualization framework, large models are often too big to fit into the available memory. The large models are cut into partitions, and the partitions are simplified and optimized individually using the available limited memory. The simplified partitions are then stitched together and used to represent the large models in the visualization. Mesh stitching also finds its application as part of other methods, especially mesh editing related areas such as mesh fusion, hole filling, and surface reconstruction as described previously in Chapter 2.

This thesis presents a novel stitching method to join two mesh partitions at their boundaries by using the curve bones. Curve approximation is lightweight compared to surface reconstruction. It will be shown in the experiments of this chapter that this method is sufficient to produce a seamless result based on some seamlessness criteria that guide the decisions made in the mesh stitching procedure.

This chapter is organized as follows. Firstly, the objectives and the outline of the proposed mesh stitching method are defined in Section 4.1 and Section 4.2, respectively. The proposed mesh stitching procedure is guided by the concept of seamlessness

Figure 4.1: (a) Two half-spheres before stitching, (b) Whole sphere after stitching, showing curve bones and stitch vertices on stitch path.

criteria, as introduced in the following Section 4.3. The full detail of the mesh stitching methodology itself is then described in Section 4.4, followed by the experimental results and discussion in Section 4.5 and conclusions in Section 4.6.

Some terms used in this chapter are defined as follows. *Stitching* is defined as the process of joining two mesh partitions together at their corresponding boundaries, while a *stitch path* is the loop of edges formed by the *stitch vertices* at which the boundaries of two mesh partitions merge (see Figure 4.1). When the term *stitch* is used alone, it refers to the *stitch path*. *Curve bones* are the constructed smooth curves that cut across the stitch path to join two mesh boundaries. The curve bones determine the positions of the stitch vertices and hence the stitch path. We use the term *across the stitch path* for the direction that goes from one partition to another side by crossing over the stitch path in the middle. We use the term *along the stitch path* for the direction that goes along on the stitch path.

4.1 Objectives

The objective of the mesh stitching is to automatically join two mesh partitions seamlessly at their corresponding boundaries. The scenario presented in the adopted large scale visualization framework requires mesh stitching for two partitions which have mismatched boundaries due to simplification or other optimization processes. The resulting stitch has to be seamless, such that it connects two partitions smoothly and the stitch is visually undetectable. Moreover, the stitch should preserve the geometry and topology of the partitions as much as possible so that they are the same as before the stitching.

4.2 Proposed Concept: Mesh Stitching Using Curve Bones

A smooth seamless surface after stitching can be imagined to consist of an unlimited number of smooth curves across the stitch. These curves are analogous to stitches that patch two pieces of clothes together in sewing. The directions of the curves are arbitrary across the stitch on the smooth surface. Some of these curves are constructed as the bones or frames of the connecting patch between two mesh partitions. This principle is used as the basic idea of seamless mesh stitching using the *curve bones* to join the mesh partitions at their boundaries.

The constructed curve bones define the joints that connect two mesh boundaries. *Stitch vertices* are then extracted from these curve bones, and together the stitch vertices form a *stitch path*. These stitch vertices replace the existing boundary vertices. The polygons connected to the boundary vertices are deleted and retriangulated to

close the gaps of partitions. The retriangulated part is the *stitch patch* (note that this term is different from the term *stitch path*).

4.3 Seamlessness Criteria

The proposed seamless stitching method in this chapter employs curve bones across the mesh partition boundaries to join the partitions together. Instead of the more geometrically accurate surface reconstruction, curve reconstruction is used. Curves are one degree lower than surfaces, hence the computation for curve reconstruction is lighter than for surface reconstruction. To complement the curve usage for the seamless stitching purpose, a set of seamlessness criteria is proposed to guide the stitching procedure to achieve a better seamless stitching result.

Seamlessness refers to the degree to which the stitch path cannot be traced or perceived visually from its surroundings. The surface around the seamless stitch should blend smoothly, while sharp feature lines across and along the stitch should be retained. In principle, a *seamless* stitch is equivalent to having a smooth transition across the stitch path without omitting the features across and along the stitch path.

The seamlessness criteria are proposed with respect to each geometric element of the mesh, that is vertex, edge, and polygon/face. In this thesis, the proposed seamlessness criteria serve to guide the construction of the steps in the proposed mesh stitching, rather than to evaluate or to measure the goal achievement. The seamlessness criteria are as follows.

4.3.1 Vertex-related criterion

The vertex-related criterion is that **the normals of vertices across the stitch are approximately continuous.**

A smooth seamless surface has vertices across the stitch with approximately continuous normals. Consequently, the curve bones across the stitch path should be constructed as smooth curves with continuous normals of the points lying on them. For that purpose, the proposed seamless mesh stitching, which will be elaborated in Section 4.4, uses Bézier curves to represent the curve bones.

If there are any feature lines across the stitch path, they should be retained; curve bones representing (close to) these feature lines should be constructed. Thus, the stitch vertices on the curve bones will form a stitch path preserving the feature lines across it.

It is unlikely that feature lines lie exactly along the stitch path, due to the tendency of the mesh previously not being cut along the feature lines to avoid the loss of the feature along the partition boundaries during any process following the cutting. However, should the feature lines lie along the stitch path, they are automatically retained by not smoothing the stitch path itself (note that only the curve *across* the stitch path is considered for smooth construction).

4.3.2 Edge-related criteria

1. The lengths of the edges across the stitch are proportional to resolutions of both mesh partitions.

In general, the *stitch patch* that joins two mesh partitions together at their boundaries should be of blended resolution between those of both partitions. To achieve the

blended resolution effect, the lengths of the edges across the stitch (edges in the *stitch patch*) should vary from the resolution or the average edge length of one partition to that of the other partition gradually.

To construct one smooth curve bone across the stitch path, two edges are chosen, each from one partition. The two boundary vertices from this edge pair will be replaced by one stitch vertex. The stitch vertex lying on the smooth curve should be at a position such that the resulting lengths of the pair of edges across the stitch path are proportionally the same as those of the edges before stitching. By maintaining the length proportion of the edges in stitching, the resolution of the polygons formed around the stitch path is blended with the surrounding mesh surface.

Uniformly spaced values of Bézier curve parameter may be easily adopted in the proposed mesh stitching to get this blended resolution effect, which will be explained in Section 4.4.3.

2. The edges with relatively large dihedral angles define the feature lines across the stitch.

The edges around the stitch can be categorized into two groups: the edges that are part of the smooth surface across the stitch, and the edges that are part of feature lines across the stitch. Edges that are part of feature lines can be identified by their relatively large dihedral angles. These lines should be preserved in stitching, hence they should be represented by some curve bones and defined by the resulting stitch vertices. Preserving feature lines across the stitch produces a seamlessly stitched model with smoothly continuous features.

4.3.3 Face-related criteria

1. The number of faces around the stitch is balanced and well distributed between the two partition meshes.

To achieve a balanced number of faces surrounding the stitch path, the stitch vertices that connect the partition boundaries should be moderate in number. A reasonable number of stitch vertices on the stitch path should lie in the range of numbers between the number of boundary vertices of each of the two partitions.

2. The sizes of faces around the stitch are approximately uniform with the surrounding surface.

Ideally, the areas of the new faces around the stitch path should not deviate much from the original surrounding faces. At the same time, to minimize geometric modification to the partitions, the proposed seamless mesh stitching limits the modification only to the first layer of boundary faces from each partition.

4.4 Stitching Methodology

The mesh stitching procedure takes in the input of two two-manifold triangle meshes (mesh partitions) each with one boundary (opening). The mesh partitions are to be joined together at their boundaries without changing the positions and orientations of the partitions relative to each other. In our large scale visualization framework, two mesh partitions are optimized without changing their positions and orientations in space, hence they do not need any registration or orientation aligning before being stitched together.

In some other related works such as mesh fusion, registration or orientation aligning often precede the joining process. For the proposed mesh stitching method, if needed, mesh partitions can be realigned using an existing registration method, such as the ICP (iterative closest point) algorithm [BM92, CM92] or its extensive variations as surveyed in [RL01], before proceeding with the proposed mesh stitching procedure.

In the proposed stitching method in this thesis, it is assumed that the pairing of boundary loops from the two partitions are known (from inference or marking); this is important when the model has been cut at its genus, producing more than one pair of boundary loops to be stitched together. In this thesis, all models used in stitching are of genus zero for the sake of simplicity. The proposed stitching method can still handle models of genus larger than zero by adopting additional checks and heuristics to determine the pairs of boundaries to be stitched together. It is also assumed the models are cut once or more to manageable sizes by parallel cuts to avoid intersecting cuts which complicate the pairing of boundaries to be matched together.

There are five sequential stages in the proposed mesh stitching method.

1. boundary band identification
2. vertex correspondence
3. curve bone construction
4. stitch vertex approximation
5. retriangulation

The workflow of the proposed mesh stitching method is illustrated in Figure 4.2.

Initially, two single outermost boundary layers of two model partitions are extracted, and the vertices of these layers are grouped into sets by a vertex correspon-

Figure 4.2: Workflow of mesh stitching using curve bones: (a) identification of boundary bands (red), (b) vertex correspondence (black dotted lines) with one sample set of vertices for curve bone construction (connected by yellow line segments), (c) construction of curve bones (one sample in orange curve) and their corresponding stitch vertices (one sample in green dot), (d) retriangulation replacing boundary bands with stitch patch (yellow area) around the stitch path (brown line).

dence rule. Each set will be used to construct a curve bone on which a stitch vertex is estimated to lie. All stitch vertices replace the existing boundary vertices from both partitions. The two boundary layers are removed and retriangulated with the stitch vertices.

Each stage of the proposed mesh stitching method is explained in detail below.

4.4.1 Boundary Band Identification

The proposed mesh stitching method initially extracts the boundary bands of mesh partitions to be joined together. Subsequently, it uses only the boundary bands to

Figure 4.3: Two boundary bands (red bands) of mesh partitions containing boundary vertices (red dots) and near-boundary vertices (blue dots).

produce the stitch patch, which minimizes the changes to the model.

The *boundary band* of a mesh partition refers to the first layer of polygons that are directly connected to the boundary vertices. A *boundary face* is a face in the boundary band. A *boundary vertex* is a vertex that corresponds to at least a pair of boundary edges, while a *boundary edge* is an edge that corresponds to only one face. Furthermore, the remaining vertices in the boundary band that are not boundary vertices are called *near-boundary vertices*. See Figure 4.3 for the illustration.

4.4.2 Vertex Correspondence

The purpose of this stage is to obtain sets of four vertices each for the curve bone construction. The four vertices comprise a pair of boundary vertices and two near-boundary vertices, one for each boundary vertex. This stage consists of two parts:

- correspondence of boundary vertices, and

- selection of representative near-boundary vertices, one for each boundary vertex

In the first part, each boundary vertex in one boundary band is paired with a boundary vertex in another boundary band. The pairing is done based on the proximity of the boundary vertices across the boundary bands. Ideally, a boundary vertex should be paired with the nearest boundary vertex of another boundary band. However, there is an order constraint of boundary vertices in the boundary band loop. A boundary vertex can only be paired to the last paired boundary vertex or the next-in-order boundary vertex at the other boundary band.

Various vertex correspondence methods [MSS92, BST00, BK02] have been suggested to solve this problem. Most methods only work effectively for two sets of vertices which commonly have similar vertex count or similarly-spaced vertices. Badly spaced vertices often have shifted correspondence (corresponding vertices are drifting apart, resulting in twirling shape of the correspondence map). Barequet and Sharir [BS95] show that the vertex correspondence problem to yield optimum pairing of vertices is indeed NP-hard (nondeterministic polynomial-time hard).

In the proposed mesh stitching procedure, vertex correspondence between two boundary bands has to be effective for two sets of vertices even with relatively different vertex counts and random spaces. A vertex correspondence rule is proposed to pair up the boundary vertices systematically by one-to-many mapping. Each boundary vertex from one boundary band is mapped to one or more next-in-list-order boundary vertices in another boundary band. Each pair of boundary vertices will participate in one curve bone construction for obtaining at least one stitch vertex at the later stage.

It is favorable for the stitch path to contain many stitch vertices from both boundary bands to retain the existing information. Nevertheless, a stitch path having as

many vertices as the total number of existing boundary vertices will produce overly crowded small polygons around the stitch path, which we want to avoid.

According to one of the seamlessness criteria in Section 4.3, a compromise is to have the stitch path to contain n_s stitch vertices, such that $n_a \leq n_s \leq n_b$, where n_a is the number of boundary vertices of boundary band a with fewer boundary vertices, and n_b is the number of boundary vertices of boundary band b with more boundary vertices. To reduce the loss of boundary information, we choose the most number of boundary vertices that can be retained, that is $n_s = n_b$. To get a stitch path of n_s stitch vertices, n_s pairs of boundary vertex correspondence must be established.

Consequently, the boundary vertex correspondence procedure is such that each boundary vertex in boundary band b corresponds to exactly one boundary vertex in boundary band a , and each boundary vertex in boundary band a corresponds to one or more consecutive boundary vertices in boundary band b .

Mismatching of the boundary vertices will lead to a poor stitching result. The criteria for an optimum match of vertices between boundaries of two mesh partitions are:

- all the distances between paired (corresponding) vertices add up to a minimum sum
- all vertices are paired up in a monotonically increasing or decreasing order in one common direction along the two boundary loops
- all vertices are paired up by one-to-one or one-to-many mappings, which are constrained by the allowable number of pairings in order to achieve a polygon distribution around the stitch, as guided by the seamlessness criteria

This vertex correspondence is an NP-hard optimization problem, in which the cost of finding the optimum match exponentially increases as the number of boundary vertices to be paired increases. The proposed stitching method thus uses a suboptimal heuristic greedy approach to pair the vertices for a reasonably good result of finding a local optimum match. The heuristic works by taking the pair of vertices with the shortest distance among several possible combinations of vertices as far as d steps forward from the current position in matching the vertices in the two boundary loops. This heuristic produces a local optimum forward match with shortest distance among local neighbor vertices. The detailed procedure of the proposed vertex correspondence is fully shown in the pseudo-code in Figure 4.4. The complexity of the vertex correspondence is $O(n_a + n_b)$, where n_a and n_b are the number of boundary vertices from each boundary band, respectively.

The vertex correspondence presented in Figure 4.4 matches vertices reasonably well in the boundaries that have uneven distribution of vertices. This can be illustrated clearly by referring to Figure 4.5 as an example of the procedure execution.

Boundary B_a is a boundary with fewer boundary vertices than those of boundary B_b , so by a one-to-many correspondence, each boundary vertex from B_a will correspond to one or more boundary vertices from B_b . Starting from an anchor pair of vertices $\mathbf{a}_1$ and $\mathbf{b}_1$ which has the shortest distance, the next set of pairs of vertices is found by sequentially pairing up $\mathbf{a}_1$ with $\mathbf{b}_{1+i}$ ($1 \leq i \leq d$), the next boundary vertices after $\mathbf{b}_1$. The sequential pairing continues for a maximum of d steps from $\mathbf{b}_1$ as long as the distance of the potential pair is less than or equal to the distance between $\mathbf{b}_{1+i}$ and $\mathbf{a}_2$, the boundary vertex next to $\mathbf{a}_1$.

```

 $B_a$ : boundary band which has  $n_a$  boundary vertices
 $B_b$ : boundary band which has  $n_b$  boundary vertices
 $n_a \leq n_b$ 
 $d$ : look forward step
 $d = n_b - n_a$ 
 $L_a[1, 2, \dots, n_a]$ : ordered list of boundary vertices of  $B_a$ 
 $L_b[1, 2, \dots, n_b]$ : ordered list of boundary vertices of  $B_b$  (same direction order as  $L_a$ )
(The indices of  $L_a$  and  $L_b$  are in cyclic order, i.e.,  $n_a + 1 = 1$ ,  $1 - 1 = n_a$ , and so on.)
 $V_C$ : ordered list of pairs of boundary vertex correspondence
 $V_C = \{\}$ 
 $k_{init}$ : index of element in  $L_b$  nearest to  $L_a[1]$ 

while  $|L_b[k_{init} - 1] - L_a[1]| \leq |L_b[k_{init} - 1] - L_a[n_a]|$ 
     $k_{init} = k_{init} - 1$ 
repeat

for every element  $L_a[t]$  where  $t = 1, 2, \dots, n_a$ 
    find as many as possible  $m$  first consecutive elements
     $\{L_b[i] | i = k_{init}, k_{init} + 1, \dots, k_{init} + m\}$ ,  $0 \leq m \leq d$ 
    where  $|L_b[i] - L_a[t]| \leq |L_b[i] - L_a[t + 1]|$ 

    put the pairs  $(L_a[t], L_b[k_{init}]), (L_a[t], L_b[k_{init} + 1]), \dots, (L_a[t], L_b[k_{init} + m])$ 
    consecutively into  $V_C$ 

     $k_{init} = k_{init} + m + 1$ 
     $d = d - m$ 

```

Figure 4.4: Pseudo-code of vertex correspondence rule.

As illustrated in Figure 4.5, the initial value of d is 5, being the difference of n_a and n_b . The value of d will decrement by the total number of forward matchings made so far. The sequential pairings are obtained by comparing distances $\mathbf{a}_1\mathbf{b}_2$ with $\mathbf{a}_2\mathbf{b}_2$, $\mathbf{a}_1\mathbf{b}_3$ with $\mathbf{a}_2\mathbf{b}_3$, $\mathbf{a}_1\mathbf{b}_4$ with $\mathbf{a}_2\mathbf{b}_4$, $\mathbf{a}_1\mathbf{b}_5$ with $\mathbf{a}_2\mathbf{b}_5$, $\mathbf{a}_1\mathbf{b}_6$ with $\mathbf{a}_2\mathbf{b}_6$, for a maximum of 5 steps forward to yield new 5 pairs of vertices, as indicated by the green links marked with number 2 in Figure 4.5, as long as the pairing distance is still less than the distance $\mathbf{a}_2\mathbf{b}_1$. If the figure is drawn to scale, the pairing should stop at $\mathbf{a}_1\mathbf{b}_3$ and the look-forward of d steps is reduced by 2 to 3. The next anchor pair will be $\mathbf{a}_2\mathbf{b}_4$ before another set of potential forward pairings as far as $d = 3$ steps is obtained.

Figure 4.5: Example of vertex correspondence for unevenly distributed boundary vertices.

This procedure will repeat until all the boundary vertices are matched in order along the boundary loops. The vertex correspondence result is shown as black links in Figure 4.5. It shows that the heuristic is able to pair up vertices well, even if the vertices are not distributed evenly across the boundary loops.

Following the boundary vertex correspondence, the second part of this stage assigns one representative near-boundary vertex to each boundary vertex. This forms a set of vertex correspondence comprising four vertices: a pair of boundary vertices together with their assigned representative near-boundary vertices. Below is the process of selecting the representative near-boundary vertices.

A boundary vertex is connected to one or more near-boundary vertices. The representative near-boundary vertex is chosen based on the dihedral angle of the edge connecting the near-boundary vertex to the corresponding boundary vertex (see Figure 4.6), according to the seamlessness criteria guide. The dihedral angle reflects the local feature of the surface. The larger the dihedral angle of an edge, the sharper or more distinct is the surface. As sharp features across the surface should be retained, the near-boundary vertex corresponding to the largest dihedral angle around

Figure 4.7: Imaginary near-boundary vertex is assigned to correspond to lone boundary vertex.

formed by the four control points, as well as its well spaced nature for each constant increment of the parameter value. These properties of a Bézier curve help create the stitching result of gradual change of resolution across the stitched partitions, as illustrated in Section 4.4.4.

Continuity at the two end points of a Bézier curve will not be much of a concern; rather, the concern will be the continuity between the boundary vertices which lie within the Bézier curve. The two end points of a Bézier curve correspond to the near-boundary vertices. These near-boundary vertices have been previously selected to complement the boundary vertex pair in representing important parts (feature or local surface condition) of the surface forming the seamless stitched surface. If these near-boundary vertices represent the feature of the surface, the surface need not be continuous around the near-boundary vertices.

The four points (two near-boundary vertices and two boundary vertices) in one vertex correspondence set are collectively used for the construction of the curve bone

Figure 4.8: Bézier curve is constructed from four control points $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$. which defines the continuity of the surface, including the two end points of the Bézier curve. More importantly, the continuity between the boundary vertices which lie within the Bézier curve is guaranteed.

The equation of the Bézier curve $\mathbf{s}(t)$ is

$$\mathbf{s}(t) = (1-t)^3\mathbf{p}_1 + 3t(1-t)^2\mathbf{p}_2 + 3t^2(1-t)\mathbf{p}_3 + t^3\mathbf{p}_4 \quad (4.1)$$

where t is the parameter of the curve such that $0 \leq t \leq 1$, $t \in \mathbb{R}$. $\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$, and $\mathbf{p}_4$ are the four control points. In this case, $\mathbf{p}_1$ is the representative near-boundary vertex corresponding to the boundary vertex $\mathbf{p}_2$, and $\mathbf{p}_4$ is the representative near-boundary vertex corresponding to the boundary vertex $\mathbf{p}_3$.

4.4.4 Stitch Vertex Approximation

According to the seamlessness criteria in Section 4.3, the stitch vertex should be located at a point such that its corresponding edges have proportional lengths matching the resolutions or average edge lengths of both sides (two mesh partitions). We will work out the stitch vertex approximation along the curve bone to produce these proportional lengths.

Let the average length of edges in the first boundary band be a and the average

Figure 4.9: Two-dimensional sketch of boundary bands (green area), a curve bone (red curve) with control points of boundary vertices (blue dots) and near-boundary vertices (yellow dots), and stitch vertices (green dots) lying on curve bone.

length of edges in the second boundary band be b , and the curve bone is to stretch from the first boundary band all the way to the second boundary band as far as L . L is the linear distance between the end points of the curve bone (see Figure 4.9 for illustration). Along the distance L , there should be x stitch vertices on the curve bone. The number of stitch vertices on the curve bone, x , and the corresponding parameter increment, Δt , can be determined by the following equations.

$$x = \lfloor \frac{L}{\frac{a+b}{2}} - 1 \rfloor \quad (4.2)$$

$$\Delta t = \frac{1}{x+1} \quad (4.3)$$

For each increment Δt for the curve bone parameter t (in Equation 4.1), a stitch vertex on the curve bone is created at the desired proportional space, creating the

Figure 4.10: Stitch vertices (green dots) are well spaced on the curve bone (red curve) with near-boundary vertices (yellow dots) as the end points, showing automatic blending of resolutions of both mesh partitions (resolutions of both partitions are represented by a simplified view of the boundary band (cyan shaded area)).

smooth blending of resolutions between both partitions (see Figure 4.10).

Given that the two mesh partitions are cut from a whole model and are not modified for their orientation and location, it can be deduced that both boundary bands are close to each other with no significant gap and no significant overlapping. Thus, $L = a + b$, $x = 1$, and $t = 0.5$. This means that for two mesh partitions obtained in the adopted large scale visualization framework, one stitch vertex at parameter value $t = 0.5$ of the curve bone will produce the desired proportional length.

The stitch path (loop) is therefore formed by all the approximated stitch vertices.

4.4.5 Retriangulation

To produce a stitch patch that joins the two mesh partitions, the retriangulation is done on both boundary bands along the stitch path. All boundary vertices are removed and replaced by the set of stitch vertices. For our mesh partitions in the adopted large scale visualization framework, each curve bone corresponds to exactly one stitch vertex. Hence, the number of stitch vertices is equal to the number of

Figure 4.11: To replace each boundary vertex $\mathbf{v}_a$ and its surrounding faces, the yellow area is retriangulated with its corners being the near boundary vertices (blue dots) of $\mathbf{v}_a$ and the corresponding boundary vertices of $\mathbf{v}_a$, that is $\mathbf{v}_{b1}$ and $\mathbf{v}_{b2}$. The remaining blue area between the two yellow areas is fitted with a triangle.

boundary vertices of one boundary band. The retriangulation for this particular boundary band is equivalent to modifying the positions of all its existing boundary vertices to the positions of all stitch vertices while retaining the topology of all the associated polygons.

For the other boundary band, the retriangulation involves removing all the existing polygons and filling up the gap. This gap can be viewed as a collection of sub-domains to be retriangulated separately. The boundaries of these sub-domains can be identified using the information obtained in the vertex correspondence stage.

To illustrate this, we define $\mathbf{v}_a$ as a boundary vertex which belongs to boundary a with fewer boundary vertices, and vertices $\mathbf{v}_{b_i}$ from boundary b as the repositioned boundary vertices which correspond to $\mathbf{v}_a$ according to the vertex correspondence information. As shown in the top yellow area in Figure 4.11, we form a sub-domain from the vertices $\mathbf{v}_{b_i}$ and the near-boundary vertices corresponding to $\mathbf{v}_a$. The sub-

Figure 4.12: Potential self-intersection retriangulation arising from a condition where a protruding triangle obstructs the vertex correspondence link between two boundary vertices.

domain is then retriangulated internally. There are as many sub-domains as the number of boundary vertices on the boundary band with fewer boundary vertices. The remaining holes are single triangles by nature (see blue area in Figure 4.11).

Hence, the general complexity of the retriangulation is $O(n)$, where n is equivalent to the total number of vertices in the two previously-existing boundary bands.

Under the worst circumstance, the nature of the boundaries to be stitched together may be so irregular that the retriangulation produces self-intersection. Figure 4.12 shows how an excessively protruding triangle (causing concave shape) along the boundary of the model obstructs the neighboring vertex correspondence, producing self-intersecting triangles after retriangulation in the stitching process. This problem can be solved by either removing such protruding triangles before stitching, or detecting the intersection between a (protruding) triangle and a correspondence line of the newly found vertex pair during the vertex correspondence stage in the mesh stitching process, and then marking it for special retriangulation to accommodate the protruding triangle on the stitch area, so that the self-intersecting retriangulation can be avoided. However, self-intersecting retriangulation could arise from other unknown

boundary configurations, and to detect and fix such self-intersecting retriangulation is an NP-hard problem [BDE96], which may involve too many checks that waste resources, especially when the models actually contain no such self-intersection (such self-intersecting cases are rarely found in this thesis experiment).

For other applications of the proposed mesh stitching to the filling of large gaps between two meshes which are far apart, the typical retriangulation methods used in hole-filling algorithms could be adopted (see Section 2.3.2 for reference on various works). In this case, each constructed curve bone will have more than one stitch vertex. These curve bones then act as the connecting frames to form the basis for the hole-filling.

4.5 Experimental Results and Discussion

4.5.1 Results

Analytically, the proposed mesh stitching method involves the processing of the vertices on two single layers of boundaries of two mesh partitions. If we approximate one unit of complexity of the method as the construction time of one curve, the complexity of the method is then equivalent to the larger number of boundary vertices of one partition, i.e., $O(n_b)$. In addition, the construction calculation of one curve is faster than that of one surface.

The proposed mesh stitching method is implemented in C++ using OpenGL and CGAL [CGA] library. An experiment is done to benchmark the performance of the program. The mesh stitching program is run on a computer with the specification of Intel Dual Core 6700 CPU at 2.66 GHz with 2 GB of RAM.

For the experiment discussed in this section, various models are represented within a normalized bounding box ranging in space from $[-1, -1, -1]$ to $[1, 1, 1]$. The original models obtained from [Aim04] and [Sta] are each cut into two partitions. These partitions are then separately simplified using MeshLab [Cig06] to either similar or different resolutions such that their boundaries are mismatched. The simplified partitions are then stitched back together to form a whole model.

The visually seamless stitching result can be observed in the various models. For models with a smooth surface across the stitch path, such as the horse model (Figure 4.13) and the camel model (Figure 4.14), we see that the vertex normals around the stitch path appear continuous, hence the surface is smooth. For models containing sharp features across the stitch path such as the Homer model (Figure 4.15), the polygons forming the mouth are stitched seamlessly: sharp features across the stitch path are retained, yet the surfaces around them are smooth; if the stitched version is compared with the directly simplified version, we see that they are almost the same (compare Figures 4.15(b) and (c)). The significance of the similarity between a stitched model and a directly simplified model is that the simplified model, obtained from the mesh cutting approach on a resource-limited system, is as reliable as the directly simplified model, which could not be obtained from a large model on the same resource-limited system (the large model has to be processed using a more powerful system to obtain the directly simplified version).

Similarly, features across the stitch path on the bumpy sphere model (Figure 4.16) and the Armadillo model (Figure 4.17) are also well retained after stitching.

Figure 4.18 shows two Homer model partitions of relatively different resolutions. The stitching result shows the seamless stitch along the boundaries of the partitions.

Figure 4.13: (a) Two simplified partitions of horse model, (b) normals of vertices around boundary bands appear approximately continuous after stitching, (c) seamlessly stitched horse.

Figure 4.14: (a) Two simplified partitions of camel model, (b) stitched camel shown with its stitch vertices and near-boundary vertices, (c) smooth looking stitched camel, (d) camel wireframe showing seamless stitch result with balanced face distribution around stitch path.

The stitch patch in the stitched Homer model looks blended in its resolution with the surrounding surfaces from the two mesh partitions.

Some versions of several models, simplified directly from the original models so that they have the same number of faces as the stitched outputs, are also provided for similarity comparison. Figure 4.19, Figure 4.20, and Figure 4.21 show the normalized distances (Hausdorff (maximum) distance, mean distance, and root mean square (RMS) distance, respectively) of two similar surfaces among the original models, the

Figure 4.15: (a) Wireframe of stitched Homer showing natural face distribution around stitch path, (b) close-up image of stitched Homer's nose and mouth area in comparison to directly simplified Homer's in (c) of same resolution.

directly simplified models, and the stitched models produced by the proposed mesh stitching method. We use the Metro tool [CRS98] for this evaluation. For a full definition of Hausdorff (maximum) distance, mean distance, and RMS distance of two surfaces, please refer to the technical report [CRS98].

The charts in Figure 4.19, Figure 4.20, and Figure 4.21 show that the stitched models are, in general, relatively close to both their original models and the directly simplified models, as all the distance values are significantly less than 1% with respect to the length of the model bounding box diagonal (insignificant difference between model surfaces). In the cases of smooth surfaces, such as the camel and the horse models, the stitched models are as good as the directly simplified models when compared with the original models in terms of the maximum distance, as seen from the Hausdorff distance of 0.606% for the camel model and 0.5009% for the horse model.

For more detailed visual results of the mesh stitching shown in the pictures here, please see the accompanying video material “[*Thesis Supplementary*] *Mesh Stitching Using Curve Bones.avi*” (Appendix B).

Figure 4.16: (a) Two simplified partitions of bumpy sphere model, (b) stitched bumpy sphere shown with its stitch vertices and near-boundary vertices, (c) stitched bumpy sphere, with seamless result along and across the stitch path, (d) wireframe of stitched bumpy sphere, which has seamless stitch path with natural face distribution around stitch path.

4.5.2 Discussion

The results above show that the proposed mesh stitching using curve bones produces visually seamless stitched models, having relatively good accuracy with respect to the original models and the models of the same resolution without stitching. In this case, the seamlessness criteria concept helps to enhance the stitch results despite the use of lightweight curve bones instead of the commonly adopted surface reconstruction approach. It also helps to retain the feature lines across the stitch.

The proposed mesh stitching procedure, guided by the seamlessness criteria, automatically works on two different mesh partitions at their boundaries assuming that mesh reorientation is not required. The proposed simple vertex correspondence rule

works relatively well in establishing the relationship between the boundaries, such that the boundary vertex pairs are well spaced with respect to each other with no vertex order overlapping. The proposed mesh stitching method only requires the input of the boundary bands of the mesh partitions and only modifies the boundary bands. By minimizing the modification to the rest of the mesh, the accuracy of the model can be largely retained after stitching. The stitch patch produced using the curve bones are naturally blended in resolution with respect to the remaining surfaces.

4.6 Contributions

This chapter presents a novel mesh stitching method using curve bones. Two boundary bands from two mesh partitions are extracted to be the input of the method. Boundary vertices and near-boundary vertices in the two boundary bands are grouped into sets of four vertices. Each set is used to construct a curve bone structure. A curve bone links two mesh partitions together. The stitch vertices lying on the curve bones collectively form a stitch path. A retriangulation is then done to replace the boundary bands with a stitch patch that joins the mesh partitions together through the stitch vertices.

The presented method produces a visually seamless stitch result without surface reconstruction, but using the more lightweight curves to define the bones or frame lines of the stitch patch joining two mesh partitions together. In order to maintain the geometric accuracy and to retain feature lines across the stitch, the proposed method is guided by a set of seamlessness criteria. The stitching is performed automatically at the boundaries of two different mesh partitions without mesh reorientation. It takes the input of only the first layer boundaries (boundary bands) for producing the stitch patch. An effective vertex correspondence rule is proposed to establish a

relationship between the boundaries given the vertex order constraint. Using Bézier curve approximation to construct the curve bone, a natural blending of the stitch patch with its surroundings can be achieved. At the same time, the curve bone mechanism is applicable to the automatic gap filling involved in joining two mesh partitions that are far apart.

The mesh stitching method presented in this chapter can be potentially applied to many fields, such as geometry modeling (for rapid modeling to join existing components to form new models), mesh editing (joining preprocessed partitions, filling holes, mesh fusion), and real time multiresolution terrain rendering.

Figure 4.17: (a) Two simplified partitions of Armadillo model, (b) stitched Armadillo shown with its stitch vertices and near-boundary vertices, (c) wireframe of stitched Armadillo (17532 faces) which has seamless stitch path and natural face distribution around stitch path, similar to wireframe of directly simplified Armadillo of same complexity (17532 faces) in (d).

Figure 4.18: (a) Two partitions of Homer model with different resolutions: 2658 faces for left partition and 674 faces for right partition, (b) stitched Homer shown with its stitch vertices and near-boundary vertices, (c) wireframe of stitched Homer which shows seamless stitch result across different resolution partitions.

Figure 4.19: Hausdorff distance comparison of original models, directly simplified models, and stitched models.

4.6. Contributions

63

Figure 4.20: Mean distance comparison of original models, directly simplified models, and stitched models.

Figure 4.21: Root mean square distance comparison of original models, directly simplified models, and stitched models.

Chapter 5

Adaptive Subpolygon Silhouette

Refinement

Real time visualization of large models often requires a lot of memory resources beyond the capacity of a standard computer system. Simplified models then substitute the large models in the visualization. To enhance the realistic appearance of the simplified models, additional details such as normals or textures are added to the simplified models in the rendering process. The texture-based simplification method, as proposed by Cohen *et al.* [COM98], makes it possible to yield real time rendering under limited computer resources. This has been an increasingly common practice in many fields, especially 3D computer games and scientific or medical visualization. While the details can be added to the inner part (polygon surfaces) of the simplified models to make the models look realistic, the low resolution simplified models are polygonal in their silhouettes. This visual artifacts affect our perception of realism of the simplified models. The first row of Figure 5.1 shows a comparison between a high resolution original model and its simplified version mapped with normal details.

Figure 5.1: Clockwise from top left: original high resolution bunny, simplified low resolution bunny with normal map details yet polygonal silhouettes, simplified low resolution bunny with refined silhouettes (without normal map details), simplified low resolution bunny with normal map details and refined silhouettes.

The simplified model has polygonal silhouettes. If these silhouettes are refined, the simplified model will be very similar to the original model.

This chapter describes in detail the proposed method to refine the polygonal silhouettes. The chapter is organized as follows. After the objective statement in Section 5.1, an overview of the methodology is given in Section 5.2. The full details of the methodology will follow in Section 5.3 and Section 5.4. The chapter will end with the experimental results and discussions in Section 5.5, and the contributions are summarized in Section 5.6.

5.1 Objectives

This chapter proposes a new real time silhouette refinement method. It aims to refine the polygonal silhouettes of low resolution models.

During runtime, the silhouettes change shape whenever the view point changes with respect to the models. As a consequence, the silhouette refinement has to take place during real time rendering. However, the refinement process should not affect the real time visualization performance.

Ideally, the polygonal silhouettes should be refined based on the fine silhouettes of the original large models. However, large model references are too big to fit into the computer memory during runtime, or the reference is not even available in the first place for applications such as 3D games. Thus, a silhouette refinement method without using large models as reference is essential.

The silhouette refinement method aims to produce better visual quality without the original model references in a resource-limited system. It refines momentarily the silhouette-related parts of the model. From the view point, the simplified models should look realistic with fine silhouettes. The result is the 3D two-manifold silhouette-refined models.

The fine silhouettes are not merely pasted as strips of polygons on top of the models, but they are integrated into the models as a two-manifold part of the models. The purpose is to present the models in more accurate form in the 3D space. It helps other space-precise visualization tasks, such as collision detection, texture and normal mapping, shading, and lighting, to maintain their accuracy in 3D space for realistic visualization.

The visualization involves viewing the model from a dynamic viewpoint. Hence, the silhouettes may change continuously, and so should the refined silhouettes. The proposed method aims to produce refined silhouettes which fit closely to the intended shape of the model. The challenge lies in how to refine the silhouette into a consistent

shape for all possible view points. If the silhouette shape is consistent (especially for a non-deforming object which does not change shape), the viewer at the view point will perceive the object shape properly without the unrealistic impression of a deforming object. Also, if the refined silhouette shape is consistent, the refined silhouettes will definitely have smooth transition upon the movement of the view point.

The proposed silhouette refinement method also aims to work adaptively with respect to the view point. It refines silhouettes to certain levels dynamically based on the viewer's visual perception of fine silhouettes at the current view point.

5.2 Methodology Overview

The silhouette refinement method takes in a low resolution 3D two-manifold polygonal model as the input. The proposed silhouette refinement methodology consists of two parts: preprocessing steps and runtime steps.

In the preprocessing stage, the input information of a model is loaded into the data structure. In order to refine the silhouettes of the model faster and also accurately during the rendering in the runtime stage, some attributes of the model are established prior to the silhouette refinement. The geometrical attributes of the model, which are independent of the view point, include the features, as well as the normals and tangents of the vertices for all facets and edges. The feature extraction and normal-tangent approximation determine the refined silhouette shape. The attribute establishment is done in the preprocessing stage to save time during runtime for real time performance.

The runtime stage is the cycle of real time processing and rendering of the model to the display. Each cycle involves the silhouette extraction given the current view point, estimation of silhouette refinement level, calculation of silhouette refinement

points, retriangulation of the silhouette parts, and rendering of the silhouette-refined model onto the display screen. This rendering result forms one frame of the display. For the next frame, if the view point position changes relative to the object, then the cycle repeats again.

Two potentially effective variants of the proposed silhouette refinement method are presented: the *silhouette edge refinement* scheme and the *subpolygon silhouette refinement* scheme. These variants differ in the effectiveness of silhouette refinement for different models with different geometry and topology. The silhouette edge refinement scheme refines the silhouette edges, while subpolygon silhouette refinement refines the subpolygon silhouette lines. Both schemes are highlighted below (compare Figure 5.2 and Figure 5.3). The full details of the preprocessing and runtime steps for each scheme will follow in Section 5.3 and Section 5.4 .

5.2.1 Silhouette Edge Refinement

The silhouette edge refinement works by extracting the silhouette edges and refining them accordingly. It consists of preprocessing and runtime stages as illustrated in the workflow shown in Figure 5.2.

The preprocessing stage starts with a 3D model being loaded into the data structure. It proceeds with a feature extraction process, where the features of the model, such as sharp points/lines/corners, are extracted based on the component connectivity characteristics. Subsequently, the normals and tangents of the vertices of the model are approximated.

During runtime, given a view point, the silhouette edges of a model are extracted. The normal and tangent information of the vertices on the silhouette edges is used

5.2. Methodology Overview

69

Figure 5.2: Workflow of silhouette edge refinement scheme.

Figure 5.3: Workflow of subpolygon silhouette refinement scheme.

to construct the smooth curves along the silhouette edges. Each curve is projected to the view plane to estimate its silhouette refinement level. The refinement level is adaptively adjusted to an optimum level at which the viewer can sufficiently perceive the silhouettes as visually fine. The refinement level indicates the number of refinement points forming the smooth curve on the silhouette edge. The refinement point positions are then calculated. The facets related to the silhouette edges are retriangulated with the refinement points.

If the view point changes with respect to the model, the whole runtime cycle will repeat again.

5.2.2 Subpolygon Silhouette Refinement

As discussed in Section 2.5, silhouette edges do not represent the actual silhouette lines which are visible on the view plane. The visible silhouette lines on the view plane are the subpolygon silhouette lines (see Figure 5.4). In addition, when the view point changes, silhouette edges may turn into non-silhouette edges, and non-silhouette edges may turn into silhouette edges. The change of silhouette status has the consequence of sudden switches between refinement and unrefinement of edges around the silhouettes. This produces some poppings (abrupt visual jumps) and gives no guarantee for smooth silhouette transition as can be seen in the silhouette edge refinement scheme.

On the other hand, subpolygon silhouette lines continuously sweep along edges of the facets of the model for any continuous movement of view point. Hence, the transition of the polygonal silhouettes is smooth. Subsequently, if the corresponding smooth curves of the subpolygon silhouette lines change continuously, then using these smooth curves to represent the refined silhouettes will also produce smooth silhouette

Figure 5.4: (a) Silhouette edges (red), (b) actual visible subpolygon silhouettes (blue).

transition.

At a given point of time, for all view points, the visible shape of refined silhouette should always correspond to exactly the same object shape in 3D space. This property of the refined silhouette is referred to as *consistency*. Further, *consistent curves* refer to curves that lie on the same smooth surface of the 3D object.

Both deformable objects and non-deformable objects may have consistent silhouette shape. Deformable objects may change shape over time, but to be consistent in shape, they only have one 3D shape at one point of time. Silhouette shape consistency is most easily observed in non-deformable objects, which do not change shape over time. A non-deformable object, that is seen as changing shape from a moving view point, is not consistent in shape. Thus, our discussion of consistent silhouette shape applies to both non-deformable and deformable objects. It is important to have the consistency property in the refined silhouette for unambiguous representation of the

model shape in space.

The silhouette properties of consistency and smooth transition are related causally. If the silhouette shape of a model is consistent for all view points, then the silhouette will change with smooth transition given a continuously moving view point. This can be reasoned as follows.

Given that the silhouette shape seen from a view point $\mathbf{C}$ is consistent with the silhouette shape seen from the nearby view point $\mathbf{C} + \Delta\mathbf{C}$ where $\Delta\mathbf{C} \rightarrow 0$, the silhouette shape must have changed by a small amount (close to zero) so the transition is continuous or smooth. Thus, given that the silhouette shape of a model is consistent for all view points, it can be deduced that the silhouette changes with smooth transition given a continuously moving view point.

In contrast, smooth transition of silhouettes does not imply silhouette shape consistency. It is noted in Section 2.5 that the subpolygon silhouette curve construction suggested by Wang *et al.* in [WTW*08] is done by using the information of only two corresponding subpolygon silhouette edges of a facet without taking into account the remaining edge on the facet. In principle, a 3D surface can only be constructed from information of at least 3 edges. The constructed subpolygon silhouette curves do not faithfully represent the underlying smooth surface of the model. For different view points, different pairs of edges of one facet produce inconsistent subpolygon silhouette curves to represent the refined silhouettes. Hence, the refined silhouettes are inconsistent in shape. Even though the refined silhouette outcome has smooth silhouette transition upon changing the view point, the refined silhouettes are not guaranteed to be consistent in shape for all view points.

A model's refined subpolygon silhouettes with consistent shapes are represented

by consistent curves lying on the refined surfaces of the model. By using the curves to represent the refined silhouettes, we can have smooth transition in silhouettes as well. This principle lays the foundation for the proposed subpolygon silhouette refinement method in this thesis.

The proposed subpolygon silhouette refinement scheme in this thesis works by approximating and refining the subpolygon silhouette lines of a model into smooth curves lying consistently on smooth surfaces of the model. The subpolygon silhouette refinement method comprises preprocessing and runtime stages as illustrated in the workflow shown in Figure 5.3.

The workflow is similar to that of the silhouette edge refinement scheme. The subpolygon silhouette refinement scheme has an additional step involving the preconstruction of Bézier patches during the preprocessing stage. The runtime stage starts with the subpolygon silhouette approximation instead of the silhouette edge extraction.

The term *preconstruct* refers to the parameterization of the Bézier surfaces without actually calculating and storing all points lying on the smooth surfaces. The term *patch* is used with equivalent meaning to the term *surface* in this chapter.

In the subpolygon silhouette refinement scheme, the geometric attributes obtained from the feature extraction and normal-tangent approximation are further processed to preconstruct the smooth Bézier patches which represent the refined facets of the low resolution model. The Bézier patches are parameterized in the preprocessing stage so that they can be used during runtime for quick calculation of the silhouette refinement points on the surfaces.

During runtime, given a view point, the subpolygon silhouette lines of a model are

extracted by the vertex visibility testing and the approximation of the subpolygon silhouette point pairs. Each subpolygon silhouette point pair forms a subpolygon silhouette line on a silhouette facet.

The subpolygon silhouette lines are projected to the view plane to determine their refinement level. Subsequently, the silhouette refinement points on the corresponding smooth curves of the subpolygon silhouette lines are calculated. The silhouette refinement points are then integrated into the polygonal model by retriangulation.

If the view point changes with respect to the model, the whole runtime cycle will repeat again.

5.3 Silhouette Edge Refinement

5.3.1 Preprocessing

The preprocessing stage of the proposed silhouette refinement method involves preliminary steps to reduce the runtime computing load of the silhouette refinement. The features of the low resolution polygonal model are extracted (Section 5.3.1.1). As the important visual parts that define the shape of the model, these features are to be retained or refined in the silhouette refinement process. As such, they dictate the surface normal and tangent approximation of the model (Section 5.3.1.2).

5.3.1.1 Feature Extraction

A feature of a model is defined as a distinctive part of the model which does not have full geometrical continuity in its locality. A feature can be a point with one or more surface curvature discontinuities or an edge which separates two smooth surfaces of

Figure 5.5: Dihedral angle corresponding to an edge in feature check is an angle between two corresponding facet normals.

different curvature.

Features have a similar role to silhouettes in defining the visible shape of a model. However, features cannot be detected by simple visibility testing or front-back face testing commonly employed in silhouette extraction schemes.

Thus, features need to be extracted separately and always be retained or refined for the runtime display together with the silhouettes. Since features of a model are static geometrical attributes (assuming the model is non-deformable), they are identified in the preprocessing stage. A feature defines the characteristic of an edge or a vertex in terms of its tangent and normals. A feature edge has tangents at its two ends pointing each other, while a feature vertex has more than one normal vector, each for a different set of facets to which it corresponds.

In the proposed silhouette refinement scheme, features are identified in the form of feature edges and feature vertices.

A feature edge is an edge which separates two smooth surfaces of different curvature. An edge is considered a feature if its two corresponding facets form a relatively acute angle, i.e., the dihedral angle (see Figure 5.5) of the facets of the edge is relatively larger than a threshold angle, which is usually set at 90° . Otherwise, it is a non-feature edge.

A feature vertex is a vertex with one or more surface curvature discontinuities around it. It is evident when it corresponds to one or more feature edges. A boundary edge, that is an edge that corresponds to only one facet, is also categorized as a feature edge. The identification of feature vertex type is summarized in Table 5.1.

Number of Connected Feature Edges	Additional Condition	Vertex Type
0	$\theta_a > \theta_{at}$	Non-feature vertex
0	$\theta_a \leq \theta_{at}$	Apex feature vertex
1		Valley feature vertex
2	$\theta_e > \theta_{et}$	Edge feature vertex
2	$\theta_e \leq \theta_{et}$	Corner feature vertex
≥ 3		Corner feature vertex

Table 5.1: Summary of feature vertex identification scheme.

In Table 5.1, θ_a is the apex angle of the vertex whose feature is to be identified, θ_{at} is the maximum threshold apex angle of an apex feature vertex, θ_e is the angle between two feature edges connected to the vertex whose feature is to be identified, and θ_{et} is the maximum threshold angle between two feature edges connected to a corner feature vertex. The apex angle of a vertex is measured as the sum of angles of facet corners immediately surrounding the vertex. The values of θ_{at} and θ_{et} are predetermined. They are usually set at 270° and 90° , respectively.

A vertex with no connected feature edge is categorized as a non-feature vertex if its apex angle is more than θ_{at} , or as an apex feature vertex if its apex angle is θ_{at} or less. A vertex with one connected feature edge is a valley feature vertex. A vertex with two or more connected feature edges is a corner feature vertex, except if the vertex has exactly two corresponding feature edges and the angle between those feature edges are larger than θ_{et} , in which case it is considered as an edge feature vertex. The illustration of various feature vertices is shown in Figure 5.6.

Figure 5.6: Vertex color code: black (non-feature vertex), yellow (apex feature vertex), red (valley feature vertex), green (edge feature vertex), blue (corner feature vertex); thick edge lines indicate feature edges. Clockwise from top left, based on feature type of vertex with black border (except black non-feature vertices): apex feature vertex, valley feature vertex, edge feature vertex, corner feature vertex.

5.3.1.2 Approximation of Normals and Tangents

For each vertex $\mathbf{v}$, the unit vertex normal $\mathbf{N}_x$ for every facet x around vertex $\mathbf{v}$ and the nearby tangent $\mathbf{t}_i$ for each edge $\mathbf{e}_i$ around vertex $\mathbf{v}$ are approximated (see Figure 5.7). The approximation is based on the feature type of the vertex and the feature type of its connected edges.

Normal and Tangent Approximation Around A Non-Feature Vertex

For a non-feature vertex $\mathbf{v}$, the unit vertex normal $\mathbf{N}_x$ for each facet x around $\mathbf{v}$ is uniform, that is the weighted average of normals $\mathbf{N}_i$ of all n facets around $\mathbf{v}$.

$$\mathbf{N}_x = \frac{\sum_{i=1}^n \theta_i \mathbf{N}_i}{|\sum_{i=1}^n \theta_i \mathbf{N}_i|} \quad (5.1)$$

where θ_i is the angle of facet corner corresponding to the vertex.

Figure 5.7: Approximation of vertex normals $\mathbf{N}_x$ for each facet x around vertex $\mathbf{v}$ and tangents $\mathbf{t}_i$ for each edge $\mathbf{e}_i$ around vertex $\mathbf{v}$.

The tangent $\mathbf{t}_i$ at $\mathbf{v}$ for each corresponding edge $\mathbf{e}_i$ around $\mathbf{v}$ is obtained from projecting the edge vector $\mathbf{e}_i$ onto a tangent plane which is perpendicular to the vertex normal $\mathbf{N}_x$ and passes through $\mathbf{v}$ (see Figure 5.8).

$$\mathbf{t}_i = \mathbf{e}_i - (\mathbf{N}_x \cdot \mathbf{e}_i) \mathbf{N}_x \quad (5.2)$$

Normal and Tangent Approximation Around An Apex Feature Vertex

For an apex feature vertex $\mathbf{v}$, the unit vertex normal $\mathbf{N}_x$ for each facet x around $\mathbf{v}$ is approximated as the normal of the facet x .

The tangent $\mathbf{t}_i$ at $\mathbf{v}$ for each corresponding edge $\mathbf{e}_i$ around $\mathbf{v}$ is fixed as the edge vector itself.

$$\mathbf{t}_i = \mathbf{e}_i \quad (5.3)$$

Figure 5.8: Approximation of tangent of edges around non-feature vertex.

Normal and Tangent Approximation Around A Valley Feature Vertex

For a valley feature vertex $\mathbf{v}$, the normal approximation is exactly the same as that of a non-feature vertex as explained previously.

The tangent at $\mathbf{v}$ for each corresponding non-feature edge around $\mathbf{v}$ is obtained in the same way as that of a non-feature vertex. The only difference is for the tangent $\mathbf{t}_{feature}$ at $\mathbf{v}$ corresponding to the feature edge, which is fixed as the feature edge vector $\mathbf{e}_{feature}$ itself.

$$\mathbf{t}_{feature} = \mathbf{e}_{feature} \quad (5.4)$$

Normal and Tangent Approximation Around An Edge Feature Vertex

The two feature edges of an edge feature vertex group the facets around them into two sides. For the edge feature vertex $\mathbf{v}$, its unit vertex normal $\mathbf{N}_x$ for each corresponding facet x on one of the two sides, is uniformly approximated as the locally weighted

average of normals $\mathbf{N}_i$ of all n facets around $\mathbf{v}$ on the same side in between two feature edges. The vertex normals for the remaining facets on the other side in between the two feature edges are uniformly approximated locally in the same way. The equation is as follows.

$$\mathbf{N}_x = \frac{\sum_{i=1}^n \theta_i \mathbf{N}_i}{|\sum_{i=1}^n \theta_i \mathbf{N}_i|} \quad (5.5)$$

where θ_i is the angle of facet corner corresponding to $\mathbf{v}$.

The tangent $\mathbf{t}_i$ at $\mathbf{v}$ for each corresponding non-feature edge $\mathbf{e}_i$ around $\mathbf{v}$ is obtained from projecting the edge vector $\mathbf{e}_i$ onto a tangent plane which is perpendicular to the local vertex normal $\mathbf{N}_x$ of its side and passes through $\mathbf{v}$ (see Figure 5.8).

$$\mathbf{t}_i = \mathbf{e}_i - (\mathbf{N}_x \cdot \mathbf{e}_i) \mathbf{N}_x \quad (5.6)$$

The feature line formed by the two feature edges connected to $\mathbf{v}$ needs to be smoothened to a curve that separates two different smooth surfaces as illustrated by the example of a cylinder edge line in Figure 5.9. The tangents $\mathbf{t}_{1\text{feature}}$ and $\mathbf{t}_{2\text{feature}}$ at $\mathbf{v}$ for the corresponding two feature edges $\mathbf{e}_{1\text{feature}}$ and $\mathbf{e}_{2\text{feature}}$, respectively, are approximated as the directed average of the feature edge vectors, as expressed in the following formulation.

$$\mathbf{t}_{1\text{feature}} = \frac{1}{2} \left(\frac{\mathbf{e}_{1\text{feature}}}{|\mathbf{e}_{1\text{feature}}|} - \frac{\mathbf{e}_{2\text{feature}}}{|\mathbf{e}_{2\text{feature}}|} \right) \quad (5.7)$$

$$\mathbf{t}_{2\text{feature}} = \frac{1}{2} \left(\frac{\mathbf{e}_{2\text{feature}}}{|\mathbf{e}_{2\text{feature}}|} - \frac{\mathbf{e}_{1\text{feature}}}{|\mathbf{e}_{1\text{feature}}|} \right) \quad (5.8)$$

Figure 5.9: Smooth edge feature curve is approximated from two feature edges around an edge feature vertex.

Normal and Tangent Approximation Around A Corner Feature Vertex

The feature edges of a corner feature vertex divide the facets around it into two or more sides. For the corner feature vertex $\mathbf{v}$, the unit vertex normal $\mathbf{N}_x$ for each facet x around $\mathbf{v}$ on one of the sides, is uniformly approximated as the locally weighted average of normals $\mathbf{N}_i$ of all n facets around $\mathbf{v}$ on the same side. The equation is similar to that of an edge feature vertex.

The tangent $\mathbf{t}_i$ at $\mathbf{v}$ for each corresponding non-feature edge around $\mathbf{v}$ is obtained in the same way as that of an edge feature vertex.

The feature line formed by the feature edges is a sharp corner feature which should be retained and not to be smoothened. The tangent $\mathbf{t}_{i_{feature}}$ at $\mathbf{v}$ for each corresponding feature edge $\mathbf{e}_{i_{feature}}$ is fixed as the feature edge vector itself.

$$\mathbf{t}_{i_{feature}} = \mathbf{e}_{i_{feature}} \quad (5.9)$$

The overall scheme for approximation of normals and tangents for all vertices of a model is summarized in Table 5.2.

Vertex Type	Connected Edge Type	Normal of the Vertex	Tangent of the Edge Related to the Vertex
Non-feature	Non-feature	Weighted average vertex normal	Projection of edge vector to normal plane
Apex feature	Non-feature	Facet normal	Edge vector
Valley feature	Feature	Weighted average vertex normal	Edge vector
Valley feature	Non-feature		Projection of edge vector to normal plane
Edge feature	Feature	Local weighted average vertex normal	Half of difference between feature edge vectors
Edge feature	Non-feature		Projection of edge vector to local normal plane
Corner feature	Feature	Local weighted average vertex normal	Edge vector
Corner feature	Non-feature		Projection of edge vector to local normal plane

Table 5.2: Summary of normal and tangent approximation.

For the silhouette edge refinement scheme, the tangent information is used in the preconstruction of the cubic Hermite curves for the silhouette edge refinement (see Figure 5.10). The Hermite curve equation $\mathbf{p}(t)$ for refining an edge is as follows:

$$\mathbf{p}(t) = \mathbf{H} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} \quad (5.10)$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{t}_1 & -\mathbf{t}_2 \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \quad (5.11)$$

Figure 5.10: Constructing Hermite curve to represent an edge based on positions of vertices and tangents at both ends.

where $\mathbf{H}$ is the matrix which is calculated in the preprocessing stage and $\mathbf{p}(t)$ is the position of a point on the Hermite curve at parameter t , $0 \leq t \leq 1$, $t \in \mathbb{R}$. The positions of the vertices corresponding to the edge are $\mathbf{v}_1$ and $\mathbf{v}_2$, while the tangents at the respective ends of the edge are $\mathbf{t}_1$ and $\mathbf{t}_2$, which are obtained in the previous normal-tangent approximation.

5.3.2 Runtime

The runtime stage of the silhouette edge refinement method involves the steps for refining the silhouettes during the runtime of the visualization. Each display frame starts with the silhouette extraction given the current view point (Section 5.3.2.1). The extracted silhouettes are then projected to the view plane to estimate their refinement level (Section 5.3.2.2), and their corresponding smooth silhouette curves are found. The smooth curves are represented by some sets of refinement points on the curves (Section 5.3.2.3), according to their levels of refinement. Following that is the silhouette retriangulation process of integrating the refinement points into the polygonal model (Section 5.3.2.4). These runtime steps are repeated every time the view point changes position with respect to the model.

5.3.2.1 Silhouette Edge Extraction

The silhouette edges of the low resolution models are extracted at runtime. The silhouettes change from one view point to another, therefore the silhouette extraction is to be done at each display frame upon changing view point. In the silhouette edge refinement scheme, silhouette edges are extracted for refinement.

There are extensive silhouette edge extraction methods to choose from depending on the purpose, as surveyed by Isenberg *et al.* in [IFH*03]. Among the various possible methods, for low resolution models in real time visualization, it is sufficiently efficient to use the edge buffer method [BS00] to extract the silhouette edges using front-and-back face testing. The proposed silhouette edge refinement method employs the edge buffer method for silhouette edge extraction in benchmarking the silhouette refinement performance in Section 5.5.

The edge buffer is a data structure which stores the silhouette status of all edges of the model. The silhouette status of an edge is obtained by comparing the two facets corresponding to the edge. If one of the facets is front-facing and another is back-facing given a view point, then the edge is a silhouette edge.

The front-and-back facing test is done for all facets of the model for each display frame. A facet is front-facing if it is facing the viewer, that is its normal vector forms an acute angle with the viewing direction vector. The facet is back-facing if it is facing away from the viewer. An edge is a silhouette edge if

$$((\mathbf{C} - \mathbf{v}) \cdot \mathbf{N}_1)((\mathbf{C} - \mathbf{v}) \cdot \mathbf{N}_2) \leq 0 \quad (5.12)$$

where $\mathbf{N}_1$ and $\mathbf{N}_2$ are the normals of the two facets corresponding to the edge, $\mathbf{C}$ is the view point, and $\mathbf{v}$ is the midpoint of the edge. The viewing direction for the edge

Figure 5.11: Front-and-back facing test for silhouette edge extraction.

is approximated as the viewing direction from the midpoint of the edge. See Figure 5.11.

In relation to the silhouette edge, the term *silhouette point* or *silhouette vertex* subsequently refers to the vertex on the silhouette edge, while *silhouette facet* refers to the one of the facets corresponding to the silhouette edge. Note that these terms are different from *subpolygon silhouette point*, *subpolygon silhouette edge*, and *subpolygon silhouette facet*, which are to be discussed in Section 5.4.

5.3.2.2 Estimation of Silhouette Refinement Level

Overall, the line segments that need to be refined by this method are all the silhouette edges and feature edges. They are projected to the view plane to measure their lengths and curvatures on the display.

The longer the projected straight line segment is, the finer the refinement is. The more curved the projected segment is (for silhouette refinement), the finer the refinement is. Higher refinement is required for a long and curved projected segment compared to that for a long and not-so-curved projected segment, and so on. The length and the curvature of the projected segment determine the refinement level. Both can be jointly represented by the area under the projected curve of the segment.

Figure 5.12: View plane projection of a refinement curve in triangle form.

We calculate the refinement level of all projected segments individually. The corresponding curve of each segment, represented by the two endpoints $\mathbf{P}_1$ and $\mathbf{P}_2$ and one curve midpoint $\mathbf{P}_3$ which all lie on the preconstructed surface, is projected onto the view plane. The three points correspond to $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, respectively, on the 2D coordinate system of the view plane. The area under the projected curve is approximated by A , that is the area (in pixel square units) of the triangle formed by the three projected points, as shown in Figure 5.12.

$$A = \frac{1}{2} |(\mathbf{p}_2 - \mathbf{p}_1) \times (\mathbf{p}_3 - \mathbf{p}_1)| \quad (5.13)$$

The area under the curve on the screen decides the refinement level: the bigger the area is, the finer the refinement is.

Next, to ensure visually continuous transition of silhouette refinement from one level to another, the refinement level is measured as a real number ranging from 1 (no refinement point in between two vertices) to approaching 0 (infinite refinement points

5.3. Silhouette Edge Refinement

87

in between two vertices). The refinement level L indicates the distance between the points on a refined segment. It is inversely proportional to the projected area of the curve on the view plane A obtained above.

$$L = A_{unit}/A \quad (5.14)$$

In the equation above, A_{unit} is the predetermined threshold value of the screen area (in pixel square units) occupied by a segment of curve in between two points for a sufficiently fine-looking refinement. The optimal value of A_{unit} can be tuned by the user (empirically) to obtain the visibly fine smooth silhouette on the screen display from any view point. This optimal value depends on several factors such as the resolution and the size of the screen display, the distance between the user and the screen display, the visual power of user's eyes, etcetera. The refinement level changes dynamically according to the curvature area on the screen which changes upon zooming in or out of the scene (see Figure 5.13). Theoretically, there are infinite silhouette refinement levels for zooming in and out regardless of the maximum resolution of the original model.

L is used as the increment value for t ($0 \leq t \leq 1$) in Equation 5.10. Based on the parameter scale t along the length of the curve segment, each point is equally spaced on the segment by a parameter increment of L to the neighboring point. The last refinement point will have the remainder space of $\frac{1}{L} - \lfloor \frac{1}{L} \rfloor$ to the next point (see Figure 5.14). Note that the arrangement does not enforce the last refinement point to be equally spaced as the former refinement points in order to establish continuous transition of refinement level for every frame in the case of relative position change of the view point.

Figure 5.13: Upon zooming in to the model, silhouette refinement level adaptively increases, so that the silhouettes still look fine. Grey shaded triangles represent same part of the model. As the model comes closer to the view plane, more polygons “grow” continuously in the direction shown by black arrows.

The number of refinement points in between two silhouette points is $\lceil \frac{1}{L} \rceil - 1$ points. The 0.5 refinement level indicates that there is 1 refinement point in between two vertices; the 0.3 refinement level indicates that there are 3 refinement points in between two vertices, with the last refinement point having the remainder space as the distance to the next neighboring point; and so on.

With regards to the potential popping problem in the silhouette transition as discussed previously in Section 5.2.2, the silhouette edge refinement scheme can be further improved by including the refinement of the non-silhouette edges that are immediately connected to the silhouette edges. The edges which have the potential to change status to silhouette edges are refined earlier before they turn into silhouette edges in the next frames. All silhouette edges and their immediate neighboring edges are projected to the view plane to estimate their refinement levels. We refer to this extended scheme as *first layer silhouette edge refinement*. The term *first layer* refers to the immediate layer of facets corresponding to the silhouette edges.

This extended scheme curbs the popping effect considerably to provide smoother silhouette transition, even though it is not completely successful and has inferior time

Figure 5.14: Silhouette refinement level and spacing between refinement points.

performance compared to the subpolygon silhouette refinement scheme as the result shows in Section 5.5.

5.3.2.3 Calculation of Silhouette Refinement Points

Silhouette refinement points on the Hermite curves are calculated based on the refinement levels of the corresponding edges. The positions of all refinement points are obtained from Equation 5.10 by using input parameter t with the increment of L , the refinement level of the edges.

The refinement points are temporarily calculated for immediate silhouette retriangulation in the next step. These points will be discarded after rendering once the view point moves relative to the model.

5.3.2.4 Silhouette Retriangulation

The silhouette retriangulation integrates the refinement points into the low resolution polygonal model to produce a two-manifold model with fine silhouettes in 3D space.

Figure 5.15: Silhouette retriangulation for silhouette edge refinement scheme.

All the refinement points on the smooth curves correspond to the existing edges. As such, the silhouette retriangulation is straightforward; the refinement points replace the existing edges to form centralized triangle fans, as shown in Figure 5.15.

When there are two or more edges containing refinement points, the silhouette retriangulation produces a triangle fan centralized at a centroid. The position of the centroid $\mathbf{c}$ is approximated based on the following formulation, which is adapted from the curved PN triangle method proposed by Vlachos *et al.* in [VPBM01].

$$\mathbf{c} = \frac{1}{6} \sum_{i=1}^6 \mathbf{T}_i = \mathbf{c}_0 + \frac{1}{18} \sum_{i=1}^3 (\mathbf{t}_{\text{start}_i} + \mathbf{t}_{\text{end}_i}) \quad (5.15)$$

where

$$\mathbf{c}_0 = \frac{1}{3} \sum_{i=1}^3 \mathbf{v}_i \quad (5.16)$$

Figure 5.16: Centroid of smooth surface formed by three Hermite curves.

and

$$\mathbf{T}_i = \begin{cases} \mathbf{v}_1 + \frac{1}{3}\mathbf{t}_{\text{start}1} & , \text{ if } i = 1 \\ \mathbf{v}_2 + \frac{1}{3}\mathbf{t}_{\text{end}1} & , \text{ if } i = 2 \\ \mathbf{v}_2 + \frac{1}{3}\mathbf{t}_{\text{start}2} & , \text{ if } i = 3 \\ \mathbf{v}_3 + \frac{1}{3}\mathbf{t}_{\text{end}2} & , \text{ if } i = 4 \\ \mathbf{v}_3 + \frac{1}{3}\mathbf{t}_{\text{start}3} & , \text{ if } i = 5 \\ \mathbf{v}_1 + \frac{1}{3}\mathbf{t}_{\text{end}3} & , \text{ if } i = 6 \end{cases} \quad (5.17)$$

$\mathbf{c}_0$ is the planar centroid of the facet. For all i , $\mathbf{v}_i$ are the vertices of the facet, $\mathbf{t}_{\text{start}_i}$ and $\mathbf{t}_{\text{end}_i}$ are the tangents at both ends of the i th edge of the facet, and $\mathbf{T}_i$ are the six control points along the edges of the silhouette facet (see Figure 5.16).

5.4 Subpolygon Silhouette Refinement

5.4.1 Preprocessing

The preprocessing stage of the subpolygon silhouette refinement scheme involves some preliminary steps similar to the silhouette edge refinement scheme. The features of the low resolution polygonal model are extracted (see previous explanation in Section 5.3.1.1), then the surface normals and tangents of the model are approximated (refer

to Section 5.3.1.2). The normal and tangent information is subsequently applied to the Bézier surface preconstruction as discussed below. The preconstructed Bézier surfaces are the underlying smooth surfaces on which the refined silhouettes lie during runtime.

Bézier Surface Parameterization

As discussed previously in Section 5.2.2, the viewer at any view point should perceive a consistent refined silhouette shape of a model. For consistent shape perception, the refined silhouettes are represented by consistent curves lying on smooth surfaces of the model at any time.

For deformable objects which have dynamic geometry attributes, the smooth surfaces change over time. For each frame at run time, the smooth surfaces of a deformable model are recalculated and the silhouettes are refined accordingly. For non-deformable objects whose geometry attributes are constant throughout the visualization, the smooth surfaces can be preconstructed in the preprocessing stage to relieve the computation burden during run time for better real time performance.

The proposed silhouette refinement method in this chapter is optimized for the input of non-deformable objects, while not preventing the input of deformable objects. As such, the smooth surface preconstruction is done in the preprocessing stage to speed up the run time for real time performance. Smooth surfaces are preconstructed by surface parameterization.

Now we discuss the surface parameterization in detail. The term *surface* is used interchangeably with the term *patch* in this context. The term *parameterization* refers to the surface preconstruction, in which the smooth surface is not actually constructed for all points lying on it; only the minimally required surface parameters

Figure 5.17: Input of Bézier surface parameterization.

are established. The surface parameterization serves its purpose at runtime for quick determination of silhouette refinement points on the surface.

We parameterize a Bézier patch based on the virtual mesh work of Su and Kumar [SK06]. The Bézier patch is in the fourth degree form, for high approximation precision of the actual silhouette shape. The patch is the base on which all subpolygon silhouette curves are to lie.

The Bézier surface parameterization for each facet involves inputs of the normals of each corresponding vertex and the tangents of both end points of all corresponding edges: three unit normal vectors and six unit tangent vectors (see Figure 5.17). The inputs are the normalized values of approximated normals and tangents found in the earlier step in Section 5.3.1.2.

In short, a point $\mathbf{p}(u, v, w)$ on a Bézier patch is determined by the following equation, which are partially calculated offline in the preprocessing stage.

$$\mathbf{p}(u, v, w) = \sum_{i+j+k=4} \mathbf{P}_{i,j,k} \frac{4!}{i!j!k!} u^i v^j w^k \quad (5.18)$$

where (u, v, w) is the barycentric coordinate in the triangle Bézier patch and for $0 \leq i \leq 4$, $0 \leq j \leq 4$, $0 \leq k \leq 4$, and $i + j + k = 4$, all $\mathbf{P}_{i,j,k}$ are the fifteen control points of the smooth surface. The barycentric parameters u , v , and w are related by $u + v + w = 1$ as two degree of freedom coordinate $(u, v, 1 - u - v)$.

The details of the calculation of the control points based on the normal and tangent information can be found in the reference [SK06]. Three of the control points at the corners are the vertices of the facet. The locations of all control points are depicted in Figure 5.18.

For surface parameterization, the fifteen control points and their related constants are precalculated and stored for runtime usage.

Figure 5.18: Bézier surface control points on a facet (orange plane): 3 vertices (red dots), 9 edge control points and 3 interior control points (black dots).

5.4.2 Runtime

The runtime stage of the subpolygon silhouette refinement scheme is similar to that of the silhouette edge refinement scheme. The runtime stage of the subpolygon silhouette refinement scheme involves the subpolygon silhouette approximation, silhouette projection to the view plane for refinement level estimation, silhouette curve construction and refinement point calculation, as well as silhouette retriangulation. Each runtime step is explained in detail below.

Figure 5.19: Two possible scenarios of different visibility status of vertices in a facet (+ is a visible vertex and - is an invisible vertex).

5.4.2.1 Subpolygon Silhouette Approximation

The subpolygon silhouette approximation is to be done at each display frame upon the changing of the view point, similar to that of the silhouette edge refinement scheme. The subpolygon silhouette approximation is done according to Hertzmann's method [HZ00]. Each vertex $\mathbf{v}$ with its normal vector $\mathbf{N}_x$ for each corresponding facet x is tested for visibility with respect to the view point $\mathbf{C}$. If the visibility value $(\mathbf{C} - \mathbf{v}) \cdot \mathbf{N}_x \geq 0$, then $\mathbf{v}$ is visible for facet x , regardless of the presence of obstacles in the line of sight between $\mathbf{v}$ and $\mathbf{C}$. A point in space with visibility $(\mathbf{C} - \mathbf{v}) \cdot \mathbf{N}_x = 0$ is a silhouette point. A feature vertex may undergo the test more than once for several vertex normals corresponding to different facets.

A facet with different visibility status among its three vertices is marked as the *subpolygon silhouette facet*. There are two possible scenarios of a subpolygon silhouette facet as shown in Figure 5.19. A subpolygon silhouette facet always has two edges with different visibility status at their ends, while another edge has the same visibility status at its ends.

In a subpolygon silhouette facet, each of its two edges which contain different visibility status at the ends, has a *subpolygon silhouette point* on it. The edge containing a subpolygon silhouette point is referred as the *subpolygon silhouette edge*. A subpoly-

Figure 5.20: Subpolygon silhouette line connects two subpolygon silhouette points on the subpolygon silhouette edges on a subpolygon silhouette facet. Subpolygon silhouette points have value of 0 in visibility value where their normals are perpendicular to the viewing direction.

gon silhouette point is a point along the edge with different visibility status where the normal at that point is perpendicular to the viewing direction. A pair of subpolygon silhouette points in a facet form a straight line called the *subpolygon silhouette line*. See Figure 5.20 for clearer depiction.

As mentioned, a subpolygon silhouette point is a point along an edge where the normal of that point is exactly perpendicular to the viewing direction vector. Approximation of the subpolygon silhouette point position at t ($0 \leq t \leq 1$) on the subpolygon silhouette edge is shown in the following vector-matrix equation, where $\mathbf{C}$ is the view point, $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{N}_1$, and $\mathbf{N}_2$ are the two vertex points and the two normals at both ends of the subpolygon silhouette edge, respectively.

$$[(1-t)(\mathbf{C} - \mathbf{v}_1) + t(\mathbf{C} - \mathbf{v}_2)]^T [(1-t)\mathbf{N}_1 + t\mathbf{N}_2] = 0 \quad (5.19)$$

5.4. Subpolygon Silhouette Refinement

97

This equation is equivalent to a quadratic equation of form $at^2 + bt + c = 0$, so that the solution of the equation is

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad (5.20)$$

where

$$a = [\mathbf{v}_1 - \mathbf{v}_2]^T [\mathbf{N}_2 - \mathbf{N}_1] \quad (5.21)$$

$$b = [\mathbf{v}_1 - \mathbf{v}_2]^T \mathbf{N}_1 + (\mathbf{C} - \mathbf{v}_1)^T [\mathbf{N}_2 - \mathbf{N}_1] \quad (5.22)$$

$$c = (\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_1 \quad (5.23)$$

For the detailed derivation of the quadratic equation above, please see Appendix A.

In practice, there is only one valid value for t which lies in the range of 0 to 1. The valid value t obtained from Equation 5.20 is used as an input for the barycentric coordinate on the preconstructed Bézier patch in Equation 5.18 to get the exact position of the subpolygon silhouette point.

The above calculation approximates the normal vector as a linear interpolation of the normals of the edge end points. This is consistent with the linear nature of the edge. In the actual refined surfaces, the normals vary at one degree lower (in equation) than the points on the surface. Hence, in reality, a common cubic surface should have quadratically varying normals, instead of linearly varying ones. Another assumption is that the viewing direction is linearly interpolated across the edge. This is also consistent with the linear nature of the edge. In the actual refined surfaces, it should be of equivalent degree to that of the smooth surface. These assumptions are adopted in order to simplify the expensive high order equation which might yield a

Figure 5.21: Refined feature edge (blue curve) bridges discontinued smooth subpolygon silhouette curves (red curves).

set of potential values of t .

In fact, only one valid t value is needed for each subpolygon silhouette edge due to the linear nature of edges of polygonal models. The visibility value of an edge may change from positive (visible) to negative (invisible), or vice versa, by crossing the visibility value of 0 only once at the subpolygon silhouette point.

In the case of the feature edge which may have two normal vector values at each of its ends, there may also be two different values of t on the edge. Consequently, the refinement along the subpolygon silhouette lines may be disconnected at the feature edges, unlike the connected subpolygon silhouette lines across the non-feature edges. At the same time, all feature edges are always refined to smooth curves. The refined feature edges act as the bridges connecting the two disconnected subpolygon silhouette curves (see Figure 5.21).

5.4.2.2 Estimation of Silhouette Refinement Level

In the subpolygon silhouette refinement scheme, the subpolygon silhouette lines are projected to the view plane to estimate their refinement levels. Some lines around the subpolygon silhouette lines also need to be projected and refined, as explained below.

If we assume that all the vertices of the low resolution model lie exactly on the

Figure 5.22: Refined subpolygon silhouette line (red curve) is visually obstructed by edges of the facets it belongs to.

original high resolution model surface, naturally, the refined subpolygon silhouette segments will not be obstructed by any vertices or unrefined parts of the model along the line of sight, except if the obstruction is actually part of a totally refined model in space. While a straight subpolygon silhouette line is not visually obstructed by the edges of the facet it belongs to, a refined subpolygon silhouette curve might be obstructed by the unrefined straight edges of the facet it belongs to as depicted in Figure 5.22. This phenomenon is possible in the situation where the curves extracted from a smooth surface coexist with the unrefined line segments on the polygonal surfaces; on a purely smooth surface, naturally the non-silhouette parts do not block the subpolygon silhouette because they are part of the same smooth surface.

We avoid the vertex perturbation in solving the obstruction problem as suggested in [WTW*08], since we assume that the position of all vertices is accurate; the vertices should remain at their own positions among the refined segments without perturbation. Indeed, it is the straight lines of edges that need further refinement to unblock the refined silhouettes.

It is observed that the most likely edges to obstruct the refined subpolygon silhouette curves are the edges of the subpolygon silhouette facets themselves. Hence,

Figure 5.23: Obstructing edge line is refined along with the subpolygon silhouette line to unblock the view of subpolygon silhouette curve representing the refined silhouette.

these edges are to be refined along with the subpolygon silhouette lines to unblock the subpolygon silhouette curves (see Figure 5.23). The subpolygon silhouette lines and the related edges are all projected to the view plane to determine their refinement levels.

We calculate the refinement level of all lines and edges as individual line segments by projection to the view plane, similar to that of the silhouette edge refinement scheme in Section 5.3.2.2.

5.4.2.3 Calculation of Silhouette Refinement Points

For each line segment to be refined, based on its refinement level determined previously, we fill up the segments with the refinement points. The refinement points are obtained by inputting their barycentric coordinates (u, v, w) into the preconstructed Bézier patch equation in Equation 5.18.

To determine the barycentric coordinates (u, v, w) of a subpolygon silhouette point, the values t_1 and t_2 for the two subpolygon silhouette points obtained in Equation

Figure 5.24: Possible arrangements of subpolygon silhouette lines with their barycentric coordinates in a facet.

5.20 are used as either u , v , or w based on the order of the subpolygon silhouette edges in the corresponding facet. Some possible arrangements are shown in Figure 5.24. The barycentric coordinates of other line segments are determined in the same way. The refinement points in between the two ends of the segments are obtained by inputting their corresponding barycentric coordinates into the Bézier patch equation in Equation 5.18.

For shading purposes, the normals $\mathbf{N}(u, v, w)$ of all the refinement points are also calculated as the cross product of two independent first derivatives $\mathbf{D}_u(u, v, w)$ and $\mathbf{D}_v(u, v, w)$ of the Bézier patch equation.

$$\mathbf{N}(u, v, w) = \frac{\mathbf{D}_u(u, v, w) \times \mathbf{D}_v(u, v, w)}{|\mathbf{D}_u(u, v, w) \times \mathbf{D}_v(u, v, w)|} \quad (5.24)$$

$$\mathbf{D}_u(u, v, w) = \frac{d\mathbf{p}(u, v, w)}{du} = \sum_{i+j+k=4} i\mathbf{P}_{i,j,k} \frac{4!}{i!j!k!} u^{i-1}v^jw^k \quad (5.25)$$

$$\mathbf{D}_v(u, v, w) = \frac{d\mathbf{p}(u, v, w)}{dv} = \sum_{i+j+k=4} j\mathbf{P}_{i,j,k} \frac{4!}{i!j!k!} u^iv^{j-1}w^k \quad (5.26)$$

The refinement points are temporarily calculated for immediate silhouette retriangulation in the next step. These points will be discarded after rendering upon the

changing of the view point in the next display frame.

5.4.2.4 Silhouette Retriangulation

The silhouette retriangulation integrates the refinement points into the low resolution polygonal model to produce a two-manifold model in 3D space.

The facets containing subpolygon silhouette lines are each retriangulated into two triangle fans: an upper triangle fan and a lower triangle fan (see Figure 5.25). The midpoints of the upper and lower triangle fans are obtained by inputting the midpoint barycentric coordinates into the preconstructed Bézier patch equation accordingly. The midpoint barycentric coordinates can be obtained by taking the middle of the corner barycentric coordinates for corresponding upper or lower parts.

The facets that contain no subpolygon silhouette line are either refined along their edges or not at all. If a facet is next to a subpolygon silhouette facet, the shared edge is refined. If an edge is a feature edge, it is refined as well. The facets containing refined edges are retriangulated in a similar way to those for the silhouette edge refinement scheme, as shown in Figure 5.15.

Figure 5.25: Silhouette retriangulation for facets containing subpolygon silhouette line.

The remaining facets without any refinement points are rendered as normal triangle polygons.

When two subpolygon silhouette segments intersect a feature edge, they intersect the feature edge at different points. As such, two cracks at these points may occur during the retriangulation. We can either insert an additional point along the border of the neighboring triangle fans across the feature edge, or patch each crack with a simple triangle connecting the subpolygon silhouette point and the surrounding pair of feature refinement points at the other side of the feature edge.

5.5 Experimental Results and Discussion

Some experiments are performed to benchmark the performance of the proposed silhouette refinement methods, that is the silhouette edge refinement scheme and its variant, first layer silhouette edge refinement, and the subpolygon silhouette refinement scheme. The silhouette refinement program is run on a computer with the specification of Intel Dual Core 6700 CPU at 2.66 GHz with 2 GB of RAM and NVIDIA Quadro FX 3500 (256 MB, 256 bits) graphics card. The display is of 1280x1024 pixel resolution and 32-bit color setting, rendering at full screen with rendered objects covering approximately 30-50% of the screen area. Each model is per-pixel shaded with Phong shading. The threshold value of A_{unit} (in Equation 5.14) for all the experiments in this chapter, unless specified differently, is fixed at a subjectively optimal value of 4.0 for all models (at this value, the author perceives the refined silhouettes as fine enough on the screen).

5.5.1 Time Performance

To benchmark the time performance of the silhouette refinement methods, four sets of experiments are performed.

In the first set of experiments, a group of relatively low resolution models are tested for various silhouette refinement schemes: non-adaptive silhouette edge refinement (as proposed in [WW03]), the proposed silhouette edge refinement scheme and its variants: the first layer silhouette edge refinement scheme and the subpolygon silhouette refinement scheme. As shown in Figure 5.26, the frame rate of the low resolution model visualization in general drops as the silhouettes of the model are refined. It is observed that all tested schemes have a similar trend.

The non-view-adaptive silhouette edge refinement is the fastest since no view projection is involved, but the silhouette will appear polygonal upon zooming in as shown in Section 5.5.2. The proposed subpolygon silhouette refinement performs better in real time than the first layer silhouette edge refinement. The silhouette edge refinement (without first layer) scheme is faster than subpolygon silhouette refinement; nevertheless the visual result of the subpolygon silhouette refinement scheme is superior to the rest of the tested schemes, as shown in the next section.

In the next experiment, various models of different resolutions are tested for their frame rate performance in the visualization with subpolygon silhouette refinement. Figure 5.27 shows that the different topology and geometry of all the models do not really affect the performance of the refinement. Rather, it is the polygon count of the model that affects the frame rate performance. The frame rate drops to the real time frame rate limit of 25 fps (frames per second) for a polygon count of around 30,000 input facets. The interactive frame rate limit of 10 fps is achieved for models with a polygon count of less than 100,000 input facets. Note that the polygon count stated in these statistics is that of the model before silhouette refinement; it does not reflect the actual total number of polygons rendered after silhouette refinement.

5.5. Experimental Results and Discussion

105

Figure 5.26: Frame rates of various models running different silhouette refinement schemes.

Figure 5.27: Frame rates of running our subpolygon silhouette refinement on various models of different resolutions.

In the third set of experiments, the performance of the proposed subpolygon silhouette refinement method is compared to that of the refinement method using PN triangles [VPBM01] with level-of-detail 1 (refined to four times the initial polygon count). Various models of different resolutions are refined using the two methods and their frame rate performances are compared. As shown in Figure 5.28, refined models with PN triangles have a better frame rate than the subpolygon-silhouette-refined models initially, but the frame rate of the former deteriorates more rapidly as the number of polygons increases. A model with approximately 30,000 input polygons, which undergoes subpolygon silhouette refinement with quartic Bézier surface parameterization, is rendered in the experiment at the real time frame rate of average 25 fps. For comparison, a scene of 30,000 PN triangles is rendered at an average frame rate of 18 fps.

While the visualization of a 30,000-polygon model, which is refined as PN triangles up to level-of-detail 1 (that is, $4 \times 30,000 = 120,000$ polygons), may have a similar or lower frame rate compared to the proposed subpolygon-silhouette-refined model of 30,000 polygons (total polygons of less than 50,000 polygons after refinement), it is important to emphasize the difference between these two approaches in terms of geometry quality of the fine silhouette produced. PN-triangle-refined models are not adaptive in their resolutions, hence upon zooming in closer to the models, the silhouettes will start to appear polygonal. The proposed subpolygon-silhouette-refined models, on the other hand, have their silhouettes adaptively refined (or unrefined) upon zooming in (or zooming out).

The final set of experiments is performed to compare the fine meshes and the subpolygon-silhouette-refined low resolution meshes which both have similarly per-

5.5. Experimental Results and Discussion

107

Figure 5.28: Performance comparison between subpolygon-silhouette-refined models and PN-triangle-refined models at different resolutions.

Threshold of subpolygon silhouette refinement, A_{unit}		10.0	7.5	5.0	2.5
Average frame rate (fps)	250-facet knot with subpolygon silhouette refinement	637.81	625.83	588.47	533.32
	High resolution knot without silhouette refinement, but with similarly-perceived silhouette quality	535.83	282.42	180.08	89.56
Polygon count of high resolution knot		5700	7600	11000	22000

Table 5.3: Comparison of frame rate performance between a silhouette-refined simplified knot (250 facets) with higher resolution knots of similarly-perceived silhouette qualities.

ceived silhouette qualities.

For each model, four pairs of the same model with different silhouette qualities are measured for their time performance. Each pair consists of two meshes of similarly perceived silhouette quality: a higher resolution fine mesh and a subpolygon-silhouette-refined lower resolution mesh. The four pairs are obtained based on the first pair of lowest silhouette quality. The first pair is obtained by choosing a subpolygon-silhouette-refined mesh at threshold $A_{unit} = 10.0$ and its higher resolution version of similarly perceived silhouette quality. The first pair is subsequently refined proportionally: the higher resolution fine mesh is further refined uniformly (by subdivision) as many times as the subpolygon silhouette refinement threshold is reduced for refining the lower resolution mesh. When this threshold is reduced by a factor of 2, the silhouette will be twice as detailed (more refinement points in the same area under a silhouette curve).

The benchmark is done for the models of knot, bunny, and horse, as shown in Table 5.3, Table 5.4, and Table 5.5, respectively. The corresponding charts are shown in Figure 5.29, Figure 5.30, and Figure 5.31.

From the tables and figures, it is observed that despite the similarly perceived

5.5. Experimental Results and Discussion

109

Figure 5.29: Time performance comparison between subpolygon-silhouette-refined knot with higher resolution knots of roughly same silhouette quality without silhouette refinement.

Threshold of subpolygon silhouette refinement, A_{unit}		10.0	7.5	5.0	2.5
Average frame rate (fps)	500-facet bunny with subpolygon silhouette refinement	566.00	550.37	536.90	489.49
	High resolution bunny without silhouette refinement, but with similarly-perceived silhouette quality	1813.05	1725.31	1534.73	293.59
Polygon count of high resolution bunny		1875	2500	3750	7500

Table 5.4: Comparison of frame rate performance between a silhouette-refined simplified bunny (500 facets) with higher resolution bunnies of similarly-perceived silhouette qualities.

silhouette quality, the time performance of rendering a higher resolution mesh generally deteriorates severely as the overall resolution of the mesh is increased compared to that of the lower resolution mesh with subpolygon silhouette refinement. This phenomenon can be accounted for from the fact that the higher resolution original mesh has to be rendered per polygon, while the subpolygon-silhouette-refined mesh will have its fewer refinement polygons rendered as triangle fans, which retains the rendering speed in general in spite of the overhead time for processing the silhouettes. This particular experiment suggests that adaptively refining the silhouettes of simplified models to achieve better visual quality is more effective (in terms of rendering

Figure 5.30: Time performance comparison between subpolygon-silhouette-refined bunny with higher resolution bunnies of roughly same silhouette quality without silhouette refinement.

Threshold of subpolygon silhouette refinement, A_{unit}		10.0	7.5	5.0	2.5
Average frame rate (fps)	600-facet horse with subpolygon silhouette refinement	365.30	364.11	354.01	353.38
	High resolution horse without silhouette refinement, but with similarly-perceived silhouette quality	1940.14	1411.07	446.02	166.37
Polygon count of high resolution horse		3000	4000	6000	12000

Table 5.5: Comparison of frame rate performance between a silhouette-refined simplified horse (600 facets) with higher resolution horses of similarly-perceived silhouette qualities.

time) as compared to merely using high resolution models.

5.5.2 Visual Result

In terms of visual quality, the quartic Bézier surface used in the proposed subpolygon silhouette refinement method yields a smooth surface with a higher degree of continuity compared to a PN triangle surface. The visual result of geometric surface error comparison between the quartic Bézier surface and the PN triangle surface in refining a low resolution hemisphere into a circular hemisphere, for example, is illustrated in Figure 5.32. It is observed that the geometric error of the PN triangle hemisphere surface is higher than that of the quartic Bézier hemisphere surface in general.

Figure 5.31: Time performance comparison between subpolygon-silhouette-refined horse with higher resolution horses of roughly same silhouette quality without silhouette refinement.

Figure 5.32: Comparison of geometric error of quartic Bézier surface and PN triangle surface of a low resolution hemisphere.

Next, this section shows the samples of visual appearance of the models refined by the proposed silhouette refinement schemes as performed in the experiments above. For complete interpretation of the visual result presented in this section, please refer to the accompanying video material “[*Thesis Supplementary*] *Subpolygon Silhouette Refinement.avi*” (Appendix B). The video shows the animation of transition of refined silhouettes upon the changing of view point under various silhouette refinement schemes.

Figure 5.33, Figure 5.34, and Figure 5.35 compare the silhouette qualities between the higher resolution original meshes with the subpolygon-silhouette-refined low resolution meshes, both having the same polygon counts, being seen from the same view

point and without change to the positions of the meshes. The change of polygonal lines of lighting effect on the surface of the knot as seen in from Figure 5.33(b) to Figure 5.33(a), as well as for the torus in Figure 5.34, reflects the structure of the silhouette refining polygons which are different from the uniformly distributed polygons in the low resolution model without silhouette refinement. However, the lighting effect in simplified low resolution models is not accurate in general, and the accurate lighting effect can be optionally produced by the commonly adopted bump mapping to correct the appearance of the surface of polygon meshes. Other information such as color, normals, etc. can also be applied onto the surface, together with the bumps, as part of the textures to be mapped onto the mesh polygons to produce more realistically lighted simplified models in real time. The bump mapping (or other additional texture mapping), which could be applied to the silhouette-refined model, is not applied to the silhouette-refined models in this chapter in order to focus on and to emphasize the silhouette refinement results. For completeness, the texture mapping is applied to case studies in Chapter 6.

The images of various silhouette refinement results are shown in Figure 5.36, Figure 5.37, Figure 5.38, Figure 5.39, Figure 5.40, and Figure 5.41. The images show that the refined silhouettes improve the realism of the low resolution models which have polygonal silhouettes. It is observed that the various proposed silhouette refinement schemes (silhouette edge refinement scheme, first layer silhouette edge refinement scheme, subpolygon silhouette refinement scheme) produce similar fine silhouette results for some models, such as Armadillo or horse.

However, the best visual results of smoothly transitioning silhouettes seen from any view points are produced by the subpolygon silhouette refinement scheme, as evident

Figure 5.33: Silhouette quality comparison: (a) 1434-facet knot without silhouette refinement, (b) 250-facet knot with subpolygon silhouette refinement (total 1434 facets).

in the accompanying video presentation. The first layer silhouette edge refinement scheme produces less poppings in the silhouette transition compared to the silhouette edge refinement scheme, but it is still inferior to the subpolygon silhouette refinement scheme.

Further, the shape consistency of the silhouette refined model is shown in the second accompanying video material “[*Thesis Supplementary*] *Subpolygon Silhouette Refinement 2.avi*” (Appendix B). It is shown that the proposed subpolygon silhouette refinement method in this thesis is able to refine the silhouettes and to retain features of a cylinder and a fandisk smoothly and consistently (the silhouette-refined cylinder and fandisk are consistent with their corresponding non-deformable shapes). By comparison, in the approach of Wang *et al.* [WTW*08], there are observed shape inconsistencies and missing feature lines.

Figure 5.42 shows how feature extraction complements the silhouette refinement in refining the silhouettes selectively and appropriately; the non-feature parts are

Figure 5.34: Silhouette quality comparison: (a) 892-facet torus without silhouette refinement, (b) 300-facet torus with subpolygon silhouette refinement (total 892 facets).

refined and the sharp features are retained carefully. The feature lines are especially well emphasized after the subpolygon silhouette refinement for the cylinder, vase, and fandisk models shown in Figure 5.43, Figure 5.44, and Figure 5.45, respectively. The view-adaptive feature of the proposed silhouette refinement schemes, as shown in Figure 5.46, helps to optimally provide refined silhouettes at any viewpoint upon zooming in or out.

5.5.3 Discussions

In general, it is observed that our subpolygon silhouette refinement method produces smooth transition of refined silhouettes upon changing view point, for both relatively low and high resolution models. The silhouette edge refinement method and the first layer silhouette refinement method have a lot of popping effects due to the status change of edges around the silhouettes from non-silhouette to silhouette status, and vice versa. This causes some parts to get refined or stop being refined in the next display frame. In the first layer refinement scheme, the first layer is only refined

Figure 5.35: Silhouette quality comparison: (a) 1324-facet bunny without silhouette refinement, (b) 500-facet bunny with subpolygon silhouette refinement (total 1324 facets).

at its edges, which are not necessarily parallel with the view plane. The viewer actually perceives the lines that are parallel to the view plane, that is the subpolygon silhouette lines, as the silhouettes. However, these subpolygon lines are not refined in this refinement scheme, so they appear irregular inside the refined edges. The irregular shapes stand out in the silhouette transition upon the changing of the view point, causing the silhouettes to look rough and popping. The subpolygon refinement method has considerably curbed these popping artifacts by adaptively refining the actually perceived subpolygon silhouette lines.

While the Bézier patch construction adapted in the proposed subpolygon silhouette refinement scheme is a more accurate way to present the smooth shape of object, it is far too expensive to calculate each of the exact subpolygon silhouette curves on the patch during runtime. Instead, a curve on the smooth preconstructed surface, corresponding to a linear subpolygon silhouette line on the silhouette facet, is drawn

between the subpolygon silhouette points as an approximation. This simplified subpolygon silhouette curve indeed produces acceptably accurate silhouettes, which are consistent in shape from all view points based on the underlying fixed surface. This silhouette curve have smooth transition upon the changing of the view point, even though the paths of the curves on the patch might slightly deviate from the actual silhouette curves on the patch.

It is observed that as the resolution of a model hits an extremely low level, the geometry information (such as vertex normals and edge tangents) of the model becomes extremely limited, and a single vertex or edge or facet may represent a significantly more complicated part of the original higher resolution model. Under such circumstances, it is probable that the normal and tangent information is inadequate to approximate the desired refined silhouette shape. The smoothness quality of the silhouette transition is deteriorating even in the subpolygon silhouette refinement scheme, due to the sudden change of status of a few long (relative to the overall size of the model) edges from subpolygon silhouette edge status to normal edge status or vice versa. The change may introduce new subpolygon silhouette lines that get refined instantly, overshadowing the smooth transition of subpolygon silhouette lines. Figure 5.47 shows an example of an extremely low resolution box with its silhouettes being refined into a smooth ellipsoid by various silhouette refinement schemes. Even though the subpolygon silhouette refinement scheme is able to successfully refine the box, the silhouette transition is still not smooth enough as shown in the accompanying video presentation (see Appendix B).

To avoid this, an extremely low resolution model could be refined to a higher resolution in preprocessing, in order to improve the shape of the preconstructed smooth

surface for the model. The previous time performance result in Section 5.5.1 suggests that the real time performance can be achieved for an input model of up to 30,000 facets (not including the additional facets after silhouette refinement), therefore using an extremely low level model to obtain fine silhouettes with the desired complex shape is not so reasonable.

In practice, for a big scene comprising of millions or billions of simplified polygons, scene graphs [Ang00] are usually used to organize the scene or models. At runtime, only the models or parts of the models that are inside the viewing frustum are loaded into the memory. Therefore, it is expected that the silhouette refinement can work sufficiently well for the objects in the viewing frustum. The proposed silhouette refinement works only in the viewing frustum. The limit figure of 30,000 polygons as the input of one scene in the viewing frustum obtained from the above experiments is therefore a sufficiently high number of polygons to handle in real time.

The proposed silhouette refinement method allows easy remapping of the inner details (normal, color, or bump (texture) maps) after the silhouette retriangulation that transforms a triangle into a triangle fan; the inner details can be interpolated accordingly (or linearly) to the additional refinement points between the existing vertices in the triangle fan. Even though this may lead to the overstretching of the texture around the refinement points, the works of [COM98, SSGH01, SM06] have suggested several ways to minimize the potential texture stretch problem.

The proposed silhouette refinement method may not be suitable for deformable objects for real time performance. Deformable objects may change shape during runtime, hence every time they change shape, the geometry attributes (features, normals, tangents) that normally can be preprocessed for silhouette refinement of non-deformable

objects have to be recalculated for every display frame. This may impede the real time performance of the visualization.

5.6 Contributions

This chapter presents a method (and its variants) to refine the polygonal silhouettes of low resolution models to complement the inner details (normal, color, bump, etcetera) in real time, given limited hardware resources (low memory, no GPU) and the unavailability of reference models during runtime.

The *silhouette edge* refinement method refines silhouette edges given a view point. On the other hand, the *subpolygon silhouette* refinement method refines the subpolygon silhouettes into smooth curves lying on the preconstructed Bézier patches, so that the refined silhouettes are closely fitted to and consistent with the shapes of the models when they are viewed from any view points. The refined parts are fitted into the models by the retriangulation into the two-manifold form. As such, the models are still suitable for other visualization processing, such as collision detection, shading, and lighting. The proposed method refines subpolygon silhouettes adaptively to produce visibly fine and smoothly transitioning silhouettes upon the changing of the view point.

The experiment has shown that the proposed subpolygon silhouette refinement method works well in the real time per-pixel rendering for refining low resolution models up to approximately 30,000 input polygons. The proposed silhouette refinement scheme properly retains sharp features and refines edge features.

In all, this chapter contributes a mechanism to refine the subpolygon silhouettes of low resolution models on the preconstructed Bézier surfaces for smooth silhou-

ette transition over the changing view point and for close-fitting, consistent shapes of silhouette refined models. The schemes of feature extraction, surface tangent and normal approximation, and optimal heuristic view-adaptive refinement level formulation as parts of the subpolygon silhouette refinement method are proposed. A suitable silhouette retriangulation method for integrating the refined silhouettes into the two-manifold models without changing the basic geometry of the models is also presented.

Successful silhouette refinement complements the visualization of low resolution models with detail maps, in substituting for the real time visualization of large models. The proposed silhouette refinement can be applied to enhancing the visual appeal in various applications, such as a 3D game, a 3D movie, and so on.

Figure 5.36: Knot (364 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.37: Torus (100 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.38: Bunny (500 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.39: Horse (600 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.40: Mannequin (699 facets) in smooth shaded and wireframe modes : (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.41: Armadillo (2000 facets): (a) low resolution with polygonal silhouettes, (b) after subpolygon silhouette refinement, (c) wireframe after subpolygon silhouette refinement.

Figure 5.42: (a) Low resolution hexagon (6 facets), (b) subpolygon-silhouette-refined hexagon without feature retention, (c) subpolygon-silhouette-refined hexagon with feature retention, (d) low resolution cylinder (86 facets), (e) subpolygon-silhouette-refined cylinder without feature retention, (f) subpolygon-silhouette-refined cylinder with feature retention.

Figure 5.43: Cylinder (86 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.44: Vase (610 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.45: Fandisk (264 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Figure 5.46: (a) Low resolution cylinder (86 facets), (b-c) smooth subpolygon-silhouette-refined cylinder viewed from a distance and its wireframe, (d-e) without view-adaptive measure, cylinder from (b) and (c) appear polygonal upon zooming in, (f-g) view-adaptive subpolygon-silhouette-refined cylinder remains smooth upon zooming in.

Figure 5.47: Extremely low resolution box (12 facets) in smooth shaded and wireframe modes: (a) low resolution with polygonal silhouettes, (b) after silhouette edge refinement, (c) after first layer silhouette edge refinement, (d) after subpolygon silhouette refinement.

Chapter 6

Case Studies

Figure 6.1: Original large Armadillo (345,944 facets)

The two main contributions of this thesis have been presented in Chapter 4 and Chapter 5. This chapter presents two case studies of the visualization of large models using their enhanced simplified version, in the real time visualization framework as de-

scribed in Chapter 3. An Armadillo model (345,944 facets) (see Figure 6.1) taken from Stanford University Computer Graphics Laboratory [Sta] and the Ramesses model (1,652,528 facets) taken from Shape Repository of AIM@SHAPE Project [Aim04] are used as the large model representatives.

In the case studies, the visualization is performed on a computer with the specification of Intel Dual Core 6700 CPU at 2.66 GHz with 2 GB of RAM and NVIDIA Quadro FX 3500 (256 MB, 256 bits) graphics card. The display is of 1280x1024 pixel resolution and 32-bit color setting. The models are per-pixel shaded with Phong shading.

Theoretically, a PC with 2 GB random access memory, which is used in our case study of large scale visualization, should be able to load up to four hundred million polygons without the presence of other running tasks that use the random access memory. In practice, as reflected by the experiment scenario in this thesis, a PC with 2 GB random access memory can handle far fewer polygons than the theoretical number, due to various factors such as additional memory allocation for data structure, program execution, and interaction with the model.

Larger models than Ramesses (1.6 million polygons) such as Lucy¹ (28.1 million polygons) could have been used for case studies, but it would be difficult to make a meaningful comparison between the enhanced simplified version and the original large models, because such a large model (Lucy) can only be displayed on the PC at extremely low frame rate (not interactive and not real-time) by a very limited number of viewer tools, which are not in the author's possession.

Even larger models, such as St. Matthew² (386.5 million polygons) that really

¹<http://graphics.stanford.edu/data/3Dscanrep/>

²<http://graphics.stanford.edu/data/dmich-public/>

Figure 6.2: (a) Two partitions of large Armadillo (213501 facets for upper partition and 133509 facets for lower partition), (b) two Armadillo partitions are simplified separately into 1230 facets for upper partition and 770 facets for lower partition (total 2000 facets before stitching).

cannot fit into the main memory even only for loading and display, could also have been used as a case study in the thesis. However, it would be a long process to partition the St. Matthew model into at least 30 parts and to optimize each of them separately, until it is ready for real time visualization on the same PC. Given the time constraint and the impossibility of comparing the result with the original large model on the same PC, this was not attempted by the author.

For practical case studies, Armadillo and Ramesses (1.6 million polygons) models are thus chosen to represent the large models for the framework presented in this thesis. From these, we infer the applicability of the framework to other larger models in general.

The original large Armadillo model is too large in size to be processed in real time with the above computer specification. Hence, the Armadillo is simplified by firstly

cutting it into two partitions as shown in Figure 6.2. The partitions are then each simplified separately in MeshLab [Cig06] using the quadric edge collapse decimation method proposed by Garland and Heckbert [GH97]. After simplification, the total number of polygons of the two partitions is 2000 facets (0.6% of original polygon count).

The partitions are then stitched together by the proposed stitching method: mesh stitching using the curve bones. The result is a unified low resolution model consisting of 2004 facets (0.2% increase of polygon count after stitching) as seen in Figure 6.3. The low resolution Armadillo is then visualized in real time, using the proposed silhouette refinement method to refine its polygonal silhouettes.

With the additional texture mapping of the surface details onto the low resolution Armadillo as shown in Figure 6.4, the appearance of the simplified Armadillo is further enhanced, resembling the original Armadillo.

A video presentation “[*Thesis Supplementary*] *Enhanced Simplified Model.avi*” (Appendix B) accompanying this thesis shows the visualization of the Armadillo model. The video shows a visualization comparison between the original large Armadillo and the low resolution Armadillo, the silhouettes of which are being refined by the proposed adaptive subpolygon silhouette refinement method. The silhouette shapes of the two versions of Armadillo in the visualizations are fine and similar to each other. On the other hand, the frame rate of the large Armadillo visualization is far below the minimum interactive frame rate (10 fps) and the minimum real time frame rate (25 fps). In contrast, the subpolygon silhouette refined Armadillo is visualized in real time beyond the minimum real time frame rate.

(a)

(b)

Figure 6.3: (a) Simplified Armadillo after stitching (2004 facets, 0.6% of original Armadillo), (b) simplified Armadillo after subpolygon silhouette refinement.

Figure 6.4: (a) Original Armadillo (345,944 facets), (b) Stitched simplified Armadillo (2004 facets) with texture (normal) map, (c) Subpolygon-silhouette-refined stitched simplified Armadillo with texture (normal) map; (d), (e), and (f) are comparisons of zoomed-in around the waist area (stitch area) of (a), (b), and (c), respectively. Silhouette-refined Armadillo with texture map looks like the original Armadillo.

Figure 6.5: Original large Ramesses (1,652,528 facets)

The other case study involving a huge polygonal model of the Ramesses is presented here as well. The original size of Ramesses model is around 1.65 million polygons as seen in Figure 6.5. One of the fastest and efficient viewers available, MeshLab [Cig06], takes 14.8 seconds to load the original Ramesses onto the computer display system used in this case study. The viewer displays the model at an average frame rate of 9.6 fps, which is below the limits of interactive frame rate (10 fps) and the real time frame rate (25 fps). Testing the model on some other available viewers on the specified computer system above causes the system to run out of memory before successfully loading it. The framework presented in this thesis makes it possible

for an off-the-shelf PC to display large models at manageable sizes (simplified models) in real time visualization, while enhancing the visual quality of the simplified models.

For the adopted visualization framework in this thesis, just like the Armadillo model, the huge Ramesses model is partitioned into two partitions. The partitions are then simplified separately and stitched together by the proposed mesh stitching using curve bones producing a simplified Ramesses model (see Figure 6.6). During visualization runtime, the simplified Ramesses with polygonal silhouettes is further enhanced by the subpolygon silhouette refinement, as shown in Figure 6.7. Using the same computer system with any viewers, most of which cannot properly load the huge original Ramesses model, the enhanced simplified Ramesses with smoothly-transitioning, fine silhouettes can be visualized at an average frame rate of 54 fps, well above the minimum real time frame rate.

The Ramesses model is further enhanced by texture details into one that resembles the original Ramesses appearance as shown in Figure 6.8.

This chapter has demonstrated the complete framework outlined in Chapter 3 for the case studies of the Armadillo and Ramesses models. The enhanced simplified models presented as the case studies in this chapter may completely substitute for the large models for the purposes of realistic real time visualization.

Figure 6.6: (a) Simplified Ramesses partitions (3999 facets for left partition and 2999 facets for right partition), (b) Simplified Ramesses after stitching (7028 facets, 0.4% of original Ramesses).

Figure 6.7: (a) Zoomed in simplified Ramesses after stitching (7,028 facets), (b) zoomed in simplified Ramesses after subpolygon silhouette refinement, (c) overall simplified Ramesses after subpolygon silhouette refinement.

Figure 6.8: (a) Original Ramesses (1,652,528 facets), (b) Stitched simplified Ramesses (7028 facets) with texture (normal) map, (c) Subpolygon-silhouette-refined stitched simplified Ramesses with texture (normal) map; (d), (e), and (f) are comparison of zoomed-in chin and shoulder area of (a), (b), and (c), respectively. Silhouette-refined Ramesses with texture map looks like the original Ramesses.

Chapter 7

Conclusions and Future Work

Large models are often too big to fit into the available memory for real time visualization. It is a common practice to simplify the large models and use lower resolution models to achieve real time visualization. However, the simplified models lack visual details and should be enhanced for better visual appeal while still achieving real time performance. The challenge is to reliably replace the large scale visualization with the real time visualization of the enhanced simplified models.

This thesis presents a real time large scale visualization framework using the enhanced simplified polygonal models. Large models are streamed into the available existing memory and cut into some manageable smaller partitions. These partitions are then optimized (simplified, remeshed, and so on) sequentially or in parallel within the available memory resource. The simplified partitions are stitched together to form a unified simplified model. During visualization at runtime, this model is enhanced by refining its silhouettes and mapping texture details onto its surfaces. The outcome is the real time visualization of a simplified model with close visual fidelity to the original large version.

7.1 Contributions

The main contributions of this thesis involve the mesh stitching and the silhouette refinement mechanisms in the visualization framework.

7.1.1 Seamless Stitching Using Curve Bones

The proposed mesh stitching method aims to join two mesh partitions at their boundaries seamlessly without reorienting the partitions. The proposed mesh stitching method uses the lightweight curve approximation instead of the computationally heavier surface reconstruction to patch the partitions together. It is possible when it is guided by the concept of seamlessness criteria.

The proposed mesh stitching method takes the input of two boundary bands, each from the two mesh partitions. The vertices in these boundary bands are grouped using a simple vertex correspondence algorithm and the seamlessness criteria to guide the decision making. The vertices in one group act as the control points to construct the curve bones. Stitch vertices lying on the curve bones are then obtained. They form a stitch path collectively. The existing two boundary bands are then replaced with a stitch patch, which is the result of the retriangulation around the boundaries to link the stitch vertices with the remaining surfaces of the mesh partitions.

The stitching method produces one complete seamlessly-stitched model. With the help of the seamlessness criteria, feature lines across the stitch are retained. The resolution of the stitch patch is blended with those of the surrounding surfaces. It is also shown that the stitched models are geometrically very close (a small distance error) to their original large version models as well as to the same models obtained from direct simplification without cutting and stitching.

The proposed mesh stitching method contains several contributions as follows.

1. Automatic stitching of two different mesh partitions at their boundaries
2. Lightweight *curve bone structure* for mesh stitching
3. *Seamlessness criteria* concept for mesh stitching
4. Minimal modification to the mesh, i.e. only the first layer boundary gets modified
5. Feature line retention across the stitch
6. Simple vertex correspondence for mesh stitching
7. Naturally blended resolution of the stitch patch between two mesh partitions with different resolutions
8. Automatic gap filling for stitching two mesh partitions that are far apart

7.1.2 Adaptive Subpolygon Silhouette Refinement

A simplified model with the texture details mapped onto it, despite the realistic look of its surface, still suffers from the polygonal silhouette artifacts. Polygonal silhouettes give the impression of a less realistic shape of the model. Therefore, the challenge is to refine the polygonal silhouettes dynamically with a smooth transition every time the view point moves relative to the model. The moving shape of the refined silhouettes should be consistent or faithful to the actual shape of the model.

The proposed adaptive subpolygon silhouette refinement method works in two stages: preprocessing and runtime. The features and geometry attributes of the model are extracted or calculated and saved in the preprocessing stage to speed up

the runtime operations. The runtime stage starts by approximating the subpolygon silhouettes for the current view point. The subpolygon silhouettes are then adaptively refined to the sufficient level that they look visibly fine at the view plane. The refinement points are incorporated into the model by retriangulation around the silhouettes. Every time the view point changes relative to the model, new sets of subpolygon silhouettes are approximated and the cycle repeats again.

The result of this adaptive subpolygon silhouette refinement method is that the silhouettes of the model look fine from any view point. The transition of the silhouettes is smooth as the view point moves around in real time. With the changing silhouettes that get refined, the shape of the model still looks consistent. The features of the model are retained or refined along with the silhouette refinement. The refined silhouettes are integral with the two-manifold geometry of the model. Up to 30,000 input polygons of low resolution models can be visualized with fine silhouettes in real time using per-pixel rendering.

The adaptive subpolygon silhouette refinement method contains several contributions as follows.

1. Subpolygon silhouette refinement on the Bézier surfaces for smooth silhouette transition and consistent silhouette shapes in representing the models faithfully
2. Feature extraction scheme for silhouette refinement
3. Surface normal and tangent approximation scheme for silhouette refinement
4. Heuristic formulation for view-adaptive silhouette refinement level
5. Infinite silhouette refinement level for zooming in and out regardless of the maximum resolution of original model if there is any

6. Silhouette retriangulation method for integrating the fine silhouettes into two-manifold models
7. Visibly-fine, real-time-refined silhouette results for low resolution models of up to 30,000 polygons input for per-pixel-lighted real time rendering

7.2 Future Work

There is some future work to improve the proposed mesh stitching method and silhouette refinement method.

The proposed mesh stitching method still has no control to prevent self-overlapping in the retriangulation (since the convex hulls of curve bones may interfere with each other) for bad or extremely irregular boundaries of mesh partitions. Possible mechanisms to solve the NP-hard problem of detecting, preventing, and fixing self-overlapping can be investigated. The stitching method can be further improved to handle more complicated features of the model across the stitch path such as folding. The proposed seamlessness criteria can be refined to use more accurate and quantifiable measurements in guiding the curve approximated stitching. Extending the mesh stitching application to filling gaps between two mesh partitions that are far apart, further investigation is required for proper retriangulation (or rather, tessellation) to fill up the in-betweens of the multiple stitch vertices at the curve bones. For this purpose, normal vectors of all stitch vertices may be approximated to produce the smooth surfaces conforming to the frames outlined by the curve bones. The mesh stitching method can be further expanded to incorporate stitching multipartitions (more than two partitions) of the model by some automatic detection of two or more boundary pairs to stitch.

The proposed silhouette refinement method can be further extended for GPU optimization in calculating and rendering the silhouette refinement points per facet. Silhouette refinement for deformable objects, which requires heavy computation during runtime for processing the silhouettes due to the dynamic geometry and topology of the object, is another area of potential research. The general effectiveness of the silhouette refinement method with smooth silhouette transition for extremely low resolution models can also be explored.

Appendix A

Derivation of Subpolygon Silhouette Point Equation

This section shows the derivation of vector-matrix equation

$$[(1-t)(\mathbf{C} - \mathbf{v}_1) + t(\mathbf{C} - \mathbf{v}_2)]^T[(1-t)\mathbf{N}_1 + t\mathbf{N}_2] = 0$$

into the quadratic equation

$$at^2 + bt + c = 0$$

where $a = [\mathbf{v}_1 - \mathbf{v}_2]^T[\mathbf{N}_2 - \mathbf{N}_1]$, $b = [\mathbf{v}_1 - \mathbf{v}_2]^T\mathbf{N}_1 + (\mathbf{C} - \mathbf{v}_1)^T[\mathbf{N}_2 - \mathbf{N}_1]$, and $c = (\mathbf{C} - \mathbf{v}_1)^T\mathbf{N}_1$.

The derivation is as follows.

148 Appendixes A. Derivation of Subpolygon Silhouette Point Equation

$$\begin{aligned}
& [(1-t)(\mathbf{C} - \mathbf{v}_1) + t(\mathbf{C} - \mathbf{v}_2)]^T [(1-t)\mathbf{N}_1 + t\mathbf{N}_2] = 0 \\
& \Leftrightarrow (1-t)^2(\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_1 + (t-t^2)((\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_2 + (\mathbf{C} - \mathbf{v}_2)^T \mathbf{N}_1) \\
& \quad + t^2(\mathbf{C} - \mathbf{v}_2)^T \mathbf{N}_2 = 0 \\
& \Leftrightarrow ((\mathbf{C} - \mathbf{v}_1)^T (\mathbf{N}_1 - \mathbf{N}_2) + (\mathbf{C} - \mathbf{v}_2)^T (\mathbf{N}_2 - \mathbf{N}_1))t^2 \\
& \quad + (-2(\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_1 + (\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_2 + (\mathbf{C} - \mathbf{v}_2)^T \mathbf{N}_1)t + (\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_1 = 0 \\
& \Leftrightarrow (\mathbf{v}_1 - \mathbf{v}_2)^T (\mathbf{N}_2 - \mathbf{N}_1)t^2 + ((\mathbf{C} - \mathbf{v}_1)^T (\mathbf{N}_2 - \mathbf{N}_1) \\
& \quad + (\mathbf{v}_1 - \mathbf{v}_2)^T \mathbf{N}_1)t + (\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_1 = 0
\end{aligned}$$

Hence,

$$a = [\mathbf{v}_1 - \mathbf{v}_2]^T [\mathbf{N}_2 - \mathbf{N}_1]$$

$$b = [\mathbf{v}_1 - \mathbf{v}_2]^T \mathbf{N}_1 + (\mathbf{C} - \mathbf{v}_1)^T [\mathbf{N}_2 - \mathbf{N}_1]$$

$$c = (\mathbf{C} - \mathbf{v}_1)^T \mathbf{N}_1$$

Appendix B

Supplementary Video

A compact disc attached to this thesis contains 4 videos titled as follows.

- [Thesis Supplementary] Mesh Stitching Using Curve Bones.avi
- [Thesis Supplementary] Subpolygon Silhouette Refinement.avi
- [Thesis Supplementary] Subpolygon Silhouette Refinement 2.avi
- [Thesis Supplementary] Enhanced Simplified Model.avi

Appendix C

Author's Publications

1. Like Gobeawan, Shuhong Xu, Stephen John Turner. *Seamless Mesh Stitching Using Curve Approximation*. In TPCG'07: Proceeding of 10th Theory and Practice of Computer Graphics 2007. Eurographics-UK Chapter. Bangor. United Kingdom. pp 55-62.

Bibliography

- [ACW*99] ALIAGA D., COHEN J., WILSON A., BAKER E., ZHANG H., ERIKSON C., HOFF K., HUDSON T., STUERZLINGER W., BASTOS R., WHITTON M., BROOKS F., MANOCHA D.: MMR: an interactive massive model rendering system using geometric and image-based acceleration. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM, pp. 199–206.
- [Aim04] AIM: AIM@SHAPE shape repository version 4.0. <http://shapes.aim-at-shape.net/index.php>, 2004. accessed on 28 February 2007.
- [Ale06] ALEXA M.: Mesh editing based on discrete Laplace and Poisson models. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM, pp. 51–59.
- [Ang00] ANGEL E.: *Interactive Computer Graphics: A Top-Down Approach with OpenGL*. Addison Wesley Longman, Inc., 2000.
- [BDE96] BAREQUET G., DICKERSON M., EPPSTEIN D.: On triangulating three-dimensional polygons. In *SCG '96: Proceedings of the twelfth annual symposium on Computational geometry* (New York, NY, USA, 1996), ACM, pp. 38–47.

-
- [BDK98] BAREQUET G., DUNCAN C. A., KUMAR S.: RSVP: A geometric toolkit for controlled repair of solid models. *IEEE Transactions on Visualization and Computer Graphics* 4, 2 (1998), 162–177.
- [BK02] BORODIN P., KLEIN R.: Progressive meshes with controlled topology modifications. In *OpenSG Symposium 2002* (January 2002).
- [BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.* 14, 2 (1992), 239–256.
- [BNK02] BORODIN P., NOVOTNI M., KLEIN R.: Progressive gap closing for mesh repairing. In *Advances in Modelling, Animation and Rendering*, Vince J., Earnshaw R., (Eds.). Springer Verlag, July 2002, pp. 201–213.
- [Boi88] BOISSONNAT J.-D.: Shape reconstruction from planar cross sections. *Comput. Vision Graph. Image Process.* 44, 1 (1988), 1–29.
- [BPK05] BISCHOFF S., PAVIC D., KOBELT L.: Automatic restoration of polygon models. *ACM Transactions on Graphics* 24, 4 (2005), 1332–1352.
- [BRM*02] BERNARDINI F., RUSHMEIER H., MARTIN I. M., MITTLEMAN J., TAUBIN G.: Building a digital model of Michelangelo’s Florentine Pieta. *Computer Graphics and Applications, IEEE* 22, 1 (Jan/Feb 2002), 59–67.
- [BS95] BAREQUET G., SHARIR M.: Filling gaps in the boundary of a polyhedron. *Computer Aided Geometric Design* 12, 2 (1995), 207–229.

-
- [BS96] BUTLIN G., STOPS C.: CAD data repair. In *Proceedings of 5th International Meshing Roundtable* (Oct 1996), Sandia National Laboratories, pp. 7–12.
- [BS00] BUCHANAN J. W., SOUSA M. C.: The edge buffer: a data structure for easy silhouette rendering. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2000), ACM, pp. 39–42.
- [BST00] BAREQUET G., SHAPIRO D., TAL A.: Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices. *The Visual Computer* 16, 2 (2000), 116–133.
- [CGA] CGAL, Computational Geometry Algorithms Library. <http://www.cgal.org>.
- [Cig06] CIGNONI P.: Meshlab version 1.1.1. <http://meshlab.sourceforge.net/>, Epoch NOE, 2006.
- [CKL07] CHONG C. S., KUMAR A. S., LEE H. P.: Automatic mesh-healing technique for model repair and finite element model generation. *Finite Elem. Anal. Des.* 43, 15 (2007), 1109–1119.
- [CM92] CHEN Y., MEDIONI G.: Object modelling by registration of multiple range images. *Image Vision Comput.* 10, 3 (1992), 145–155.
- [CMRS98] CIGNONI P., MONTANI C., ROCCHINI C., SCOPIGNO R.: A comparison of mesh simplification algorithms. *Computers & Graphics* 22, 1 (1998), 37–54.

-
- [CMRS03] CIGNONI P., MONTANI C., ROCCHINI C., SCOPIGNO R.: External memory management and simplification of huge meshes. *IEEE Transactions on Visualization and Computer Graphics* 9, 4 (2003), 525–537.
- [CMSR98] CIGNONI P., MONTANI C., SCOPIGNO R., ROCCHINI C.: A general method for preserving attribute values on simplified meshes. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 59–66.
- [COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 115–122.
- [CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum* 17, 2 (1998), 167–174.
- [CW04] CHEN S.-G., WU J.-Y.: Estimating normal vectors and curvatures by centroid weights. *Comput. Aided Geom. Des.* 21, 5 (2004), 447–458.
- [dCS96] DE COUGNY H. L., SHEPHARD M. S.: Surface meshing using vertex insertion. In *Proceedings of the 5th International Meshing Roundtable* (1996), pp. 243–256.
- [Die93] DIERCKX P.: *Curve and surface fitting with splines*. Oxford University Press, Inc., New York, NY, USA, 1993.

-
- [DRS08] DYKEN C., REIMERS M., SELAND J.: Real-time GPU silhouette refinement using adaptively blended Bèzier patches. *Computer Graphics Forum* 27, 1 (2008), 1–12.
- [FKS*04] FUNKHOUSER T., KAZHDAN M., SHILANE P., MIN P., KIEFER W., TAL A., RUSINKIEWICZ S., DOBKIN D.: Modeling by example. *ACM Trans. Graph.* 23, 3 (2004), 652–663.
- [FKU77] FUCHS H., KEDEM Z. M., USELTON S. P.: Optimal surface reconstruction from planar contours. *SIGGRAPH Comput. Graph.* 11, 2 (1977), 236–236.
- [FSSW07] FOSTER K., SOUSA M. C., SAMAVATI F. F., WYVILL B.: Polygonal silhouette error correction: A reverse subdivision approach. *International Journal of Computational Science and Engineering* 3, 1 (2007), 53–70.
- [GGH*99] GU X., GORTLER S. J., HOPPE H., MCMILLAN L., BROWN B. J., STONE A. D.: *Silhouette Mapping*. Tech. Rep. TR-1-99, Harvard University, Computer Science Technical Report, 1999.
- [GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.
- [GS02] GARLAND M., SHAFFER E.: A multiphase approach to efficient surface simplification. In *VIS '02: Proceedings of the conference on Visualization*

- '02 (Washington, DC, USA, 2002), IEEE Computer Society, pp. 117–124.
- [GS03] GARIMELLA R. V., SWARTZ B. K.: *Curvature Estimation for Unstructured Triangulations of Surfaces*. Technical Report LA-UR-03-8240, Los Alamos National Laboratory, 2003.
- [GTLH01] GUÉZIEC A., TAUBIN G., LAZARUS F., HORN B.: Cutting and stitching: Converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics* 7, 2 (2001), 136–151.
- [HLK01] HO J., LEE K. C., KRIEGMAN D.: Compressing large polygonal models. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 357–362.
- [Hop96] HOPPE H.: Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 99–108.
- [Hop98] HOPPE H.: Smooth view-dependent level-of-detail control and its application to terrain rendering. In *VIS '98: Proceedings of the conference on Visualization '98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 35–42.
- [HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 517–526.

- [IFH*03] ISENBERG T., FREUDENBERG B., HALPER N., SCHLECHTWEIG S., STROTHOTTE T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Comput. Graph. Appl.* 23, 4 (2003), 28–37.
- [IG03] ISENBURG M., GUMHOLD S.: Out-of-core compression for gigantic polygon meshes. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 935–942.
- [ILGS03] ISENBURG M., LINDSTROM P., GUMHOLD S., SNOEYINK J.: Large mesh simplification using processing sequences. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 61.
- [JLW*06] JIN X., LIN J., WANG C. C. L., FENG J., SUN H.: Mesh fusion using functional blending on topologically incompatible sections. *Vis. Comput.* 22, 4 (2006), 266–275.
- [KSMK99] KANAI T., SUZUKI H., MITANI J., KIMURA F.: Interactive mesh fusion based on local 3D metamorphosis. In *Proceedings of the 1999 conference on Graphics interface '99* (San Francisco, CA, USA, 1999), Morgan Kaufmann Publishers Inc., pp. 148–156.
- [Lie03] LIEPA P.: Filling holes in meshes. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 200–205.
- [Lin00] LINDSTROM P.: Out-of-core simplification of large polygonal models. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Com-*

- puter graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 259–262.
- [LLS*04] LEE Y., LEE S., SHAMIR A., COHEN-OR D., SEIDEL H.-P.: Intelligent mesh scissoring using 3D snakes. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 279–287.
- [LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital Michelangelo project: 3D scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 131–144.
- [LS01] LINDSTROM P., SILVA C. T.: A memory insensitive technique for large model simplification. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 121–126.
- [LSTS04] LI Y., SUN J., TANG C.-K., SHUM H.-Y.: Lazy snapping. *ACM Trans. Graph.* 23, 3 (2004), 303–308.
- [Lue01] LUEBKE D. P.: A developer's survey of polygonal simplification algorithms. *IEEE Comput. Graph. Appl.* 21, 3 (2001), 24–35.
- [LZHM06] LAI Y.-K., ZHOU Q.-Y., HU S.-M., MARTIN R. R.: Feature sensitive mesh segmentation. In *SPM '06: Proceedings of the 2006 ACM sym-*

- posium on Solid and physical modeling* (New York, NY, USA, 2006), ACM, pp. 17–25.
- [Max99] MAX N.: Weights for computing vertex normals from facet normals. *J. Graph. Tools* 4, 2 (1999), 1–6.
- [MSS92] MEYERS D., SKINNER S., SLOAN K.: Surfaces from contours. *ACM Trans. Graph.* 11, 3 (1992), 228–258.
- [MW99] MANGAN A. P., WHITAKER R. T.: Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics* 5, 4 (1999), 308–321.
- [NBM05] NILSSON O., BREEN D., MUSETH K.: Surface reconstruction via contour metamorphosis: an Eulerian approach with Lagrangian particle tracking. In *VIS '05: Proceedings of the 16th IEEE Visualization 2005 (VIS'05)* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 407–414.
- [OPC96] OLIVA J.-M., PERRIN M., COQUILLART S.: 3D reconstruction of complex polyhedral shapes from contours using a simplified generalized Voronoi diagram. *Computer Graphics Forum* 15, 3 (1996), 397–408.
- [PC01] PETERSSON N. A., CHAND K. K.: Detecting translation errors in CAD surfaces and preparing geometries for mesh generation. In *Proceedings of 10th International Meshing Roundtable* (2001), Sandia National Laboratories, pp. 63–371.

-
- [Pet02] PETITJEAN S.: A survey of methods for recovering quadrics in triangle meshes. *ACM Comput. Surv.* 34, 2 (2002), 211–262.
- [PK96] PARK H., KIM K.: Smooth surface approximation to serial cross-sections. *Computer-Aided Design* 28, 12 (1996), 995–1005.
- [Pri00] PRINCE C.: *Progressive Meshes for Large Models of Arbitrary Topology*. Master's thesis, University of Washington, Seattle, USA, 2000.
- [PS96] PFEIFLE R., SEIDEL H.-P.: Triangular B-splines for blending and filling of polygonal holes. In *GI '96: Proceedings of the conference on Graphics interface '96* (Toronto, Ont., Canada, Canada, 1996), Canadian Information Processing Society, pp. 186–193.
- [RL00] RUSINKIEWICZ S. M., LEVOY M.: QSpIat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 343–352.
- [RL01] RUSINKIEWICZ S. M., LEVOY M.: Efficient variants of the ICP algorithm. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling* (2001), pp. 145–152.
- [Sal05] SALOMON D.: *Curves and Surfaces for Computer Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

-
- [SCESL] SILVA C. T., CHIANG Y.-J., EL-SANA J., LINDSTROM P.: Out-of-core algorithms for scientific visualization and computer graphics. In *Visualization '02 Course Notes*.
- [SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM, pp. 175–184.
- [SG01] SHAFFER E., GARLAND M.: Efficient adaptive simplification of massive meshes. In *VIS '01: Proceedings of the conference on Visualization '01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 127–134.
- [SGG*00] SANDER P. V., GU X., GORTLER S. J., HOPPE H., SNYDER J.: Silhouette clipping. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 327–334.
- [SK06] SU Y., KUMAR A. S.: Templatized refinement of triangle meshes using surface interpolation. *Int. J. Numer. Methods Eng.* 65, 9 (2006), 1472–1494.
- [SM06] SANDER P. V., MITCHELL J. L.: Progressive buffers: view-dependent geometry and texture LOD rendering. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM, pp. 1–18.
- [SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *SIGGRAPH '01: Proceedings of the*

28th annual conference on Computer graphics and interactive techniques

(New York, NY, USA, 2001), ACM, pp. 409–416.

[Sta] The stanford 3D scanning repository. <http://www-graphics.stanford.edu/data/3Dscanrep/>.

[SWC00] STEINBRENNER J. P., WYMAN N. J., CHAWNER J. R.: Fast surface meshing on imperfect CAD models. In *Proceedings of 9th International Meshing Roundtable* (2000), pp. 33–41.

[TL94] TURK G., LEVOY M.: Zippered polygon meshes from range images. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1994), ACM, pp. 311–318.

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved PN triangles. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (New York, NY, USA, 2001), ACM, pp. 159–166.

[WARV02] WEISS V., ANDOR L., RENNER G., VÁRADY T.: Advanced surface fitting techniques. *Comput. Aided Geom. Des.* 19, 1 (2002), 19–42.

[WTD04] WANG A. Y., TANG M., DONG J. X.: A survey of silhouette detection techniques for non-photorealistic rendering. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics (ICIG'04)* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 434–437.

- [WTW*08] WANG L., TU C., WANG W., MENG X., CHAN B., YAN D.: Silhouette smoothing for real-time rendering of mesh surfaces. *IEEE Transactions on Visualization and Computer Graphics* 14, 3 (2008), 640–652.
- [WW03] WANG B., WANG W.: Silhouette smoothing by boundary curve interpolation. In *CAD/Graphics '03: Proceedings of 8th International Conference on CAD/Graphics* (Taipa, Macao, China, 2003), IEEE Computer Society, pp. 197–202.
- [WWWG06] WU H., WEI L.-Y., WANG X., GUO B.: Silhouette texture. In *Rendering Techniques '06 (Proc. of the Eurographics Symposium on Rendering)* (June 2006), The European Association for Computer Graphics, pp. 285–296.
- [YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with Poisson-based gradient field manipulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 644–651.